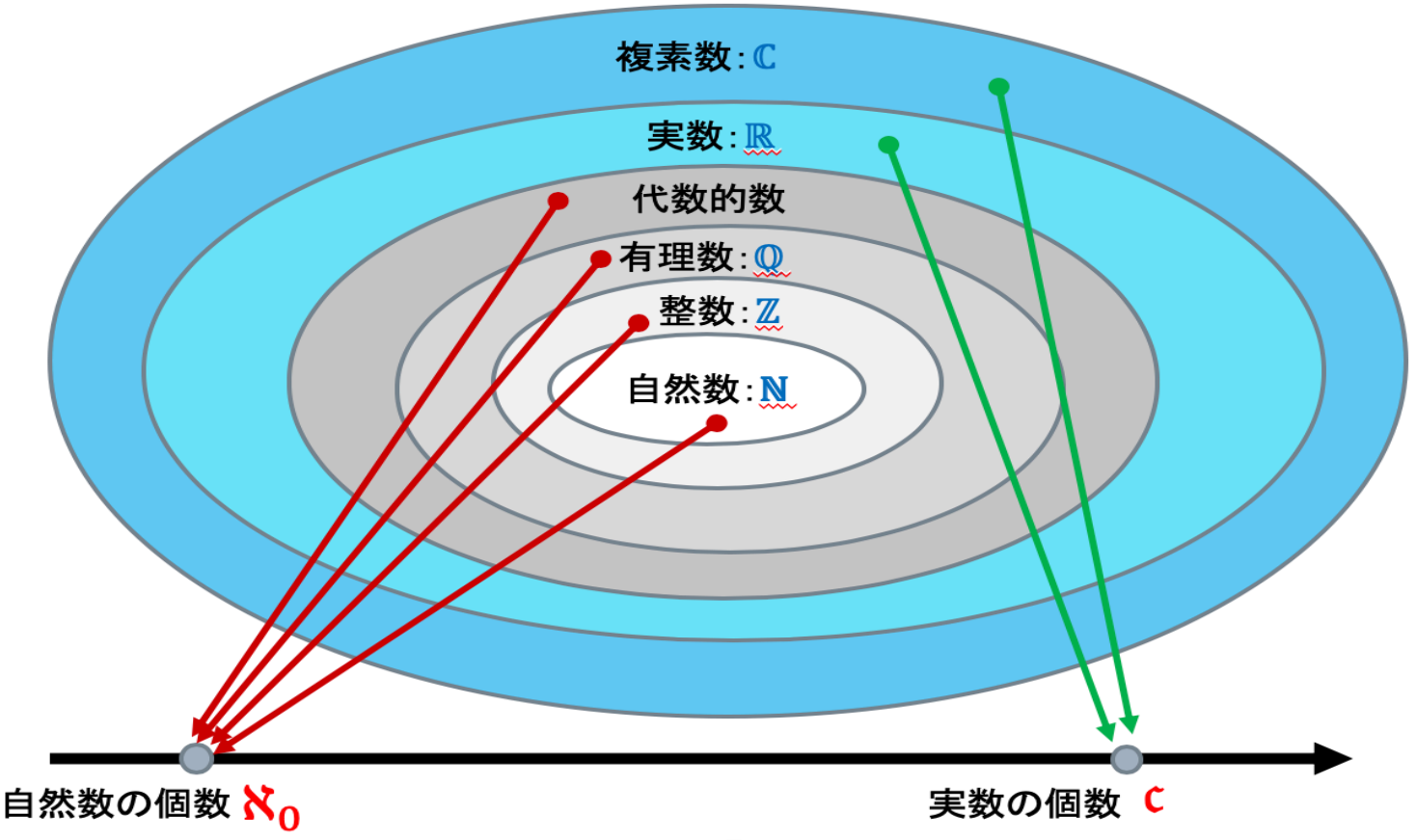
2のn乗の話

$\mathfrak{c} = 2^{\aleph_0}$?



Agenda
2のn乗の話

I. 「2のn乗」の様々な解釈

II. $\mathfrak{c} = 2^{\aleph_0}$ ：連続体仮説

III. Turing Machine入門

IV. Turing Machineと計算可能な数

Agenda
「2のn乗」の様々な解釈

- 2のn乗の基本的な解釈
  - 2のn乗と二進数
  - 二つのものを重複を許してn個並べる順列の数
  - n個の要素を持つ集合の、部分集合の数
- 2のn乗は関数を表す

Agenda

$\mathfrak{c} = 2^{\aleph_0}$: 連続体仮説

- 「個数を数える」を分析する
  - 数えることと個数
- 無限集合の要素の個数 -- カントールが考えたこと
  - 無限集合の不思議 -- 部分は、全体と同じ数の要素を持つ
  - ヒルベルトの無限ホテル
- 0と1を結ぶ直線上の点の数
  - 平面上の点の数
  - カントール集合：奇妙な図形上の点の数
  - 不思議で奇妙な例
- カントールが証明できなかったこと --「連続体仮説」
- 非カントール的集合論の発見
  - 連続体仮説と集合論の無矛盾性の証明
  - 連続体仮説と集合論の独立性の証明

Agenda
Turing Machine入門

- チューリング・マシンとは何か？
- チューリング・マシンのサンプル
  - 例1： 文字列の長さを求める
  - 例2： 文字列中の'1'の数の偶奇を求める
  - 例3： 文字列のコピー
  - 例4： 括弧のバランスのチェック
- チューリング・マシンとコンピュータのプログラム
  - 例5： 無限ループ
  - 例6： Goto L1, Goto L2
  - 例7： 条件分岐

Agenda
Turing Machineと計算可能な数

- チューリング・マシンと言語理論
- チューリング・マシンで計算できる数
- チューリング・マシンと停止問題
- 計算可能な自然数上の関数の個数
  - 乱数列
  - Conwayの「自由意志定理」
- 計算可能だが、とても「計算が難しい」ものがあること
  - アッカーマン関数
  - Busy Beaver 問題
- 複雑性理論へ – nと2^n

「2のn乗」の様々な解釈

Agenda
「2のn乗」の様々な解釈

- 2のn乗の基本的な解釈
  - 2のn乗と二進数
  - 二つのものを重複を許してn個並べる順列の数
  - n個の要素を持つ集合の、部分集合の数
- 2のn乗は関数を表す

2のn乗の基本的な解釈

2のn乗と二進数

2のn乗と二進数

- IT の世界の人は、1, 2, 4, 8, 16, 32, 64, ... という2のn乗の並びはよく知っているとおもう。ここでは、まず、皆も知っている2進数の話をしよう。
- 三桁の二進数$b_2b_1b_0$を十進法で表すにはどうすればいいか考えてみよう。ただし、b_iは 0か1で、たとえば、二進数が 101 なら、$b_2=1$, $b_1=0$, $b_0=1$ ということにする。
- $101 = 1x2^2 + 0x2^1 + 1x2^0 = 4 + 0 + 1 = 5$ である。
  $b_2b_1b_0 = b_2x2^2 + b_1x2^1 + b_0x2^0$

- 一般に、
  $b_nb_{n-1}...b_1b_0 = b_nx2^n + b_{n-1}x2^{n-1} + ... + b_1x2^1 + b_0x2^0$

1より小さい数を二進数で表す

- 二進数で、1より小さい数を表すことができる。
  小数点 '.' の後ろに、0と1を並べればいい。
- たとえば、
  $.1 = 1\times2^{-1} = 1\times(1/2) = 0.5$
  $.101 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = 1/2 + 1/8$
- 一般に、小数点以下 i 番目の数字を b_{-i} で表すと、
  $.b_{-1}b_{-2}\ldots b_{-n} = b_{-1}\times2^{-1} + b_{-2}\times2^{-2} + \ldots + b_{-n}\times2^{-n}$
- もっと一般に、 小数点
  $b_nb_{n-1}\ldots b_1b_0 . b_{-1}b_{-2}\ldots b_{-m}$
  $= b_n\times2^n + b_{n-1}\times2^{n-1} + \ldots + b_1\times2^1 + b_0\times2^0$
  $+ b_{-1}\times2^{-1} + b_{-2}\times2^{-2} + \ldots + b_{-m}\times2^{-m}$
  $$= \sum_{i=-m}^{n} b_i 2^i$$

数字と文字列

- 今まで見てきた二進数は、数を表すものとして解釈されてきた。例えば、101は、数 5を表すと。
- ただ、101を十進法で <ひゃくいち>を表すと解釈することも、三進法で<10>を表すと解釈することも可能である。また、ある場合には、それを、文字列"101"と解釈することも可能である。どの解釈が選ばれているかは、コンテキストで暗黙のうちに決められていることが多い。
- もっとも汎用的なのは、数字を表す「文字」があらかじめ定義されていて、それらの文字を連ねて作られる「文字列」が、数字を表すと考えることである。例えば、

2進数を表す文字　＝ { '0', '1' }
16進数を表す文字 ＝ { '0', '1', '2', '3', '4', '5', '6', '7',
'8', '9', 'A', 'B', 'C', 'D', 'E', 'F' }

数字と「語の列」

- 数字を表す「文字」があらかじめ定義されていて、それらの文字を連ねて作られる「文字列」が数字を表すという話をしたが、自然言語では、必ずしもそうではない。
- 自然言語では、数字を表す「語」があらかじめ定義されていて、それらの語を連ねて作られる「語の列」が数字を表す。
- 例えば、日本語では、次の語の列が数字を表す。
  { “いち”, “に”, “さん”, “し”, “ご”, “ろく”, “なな”,
  “はち”, “く”, “じゅう”, “ひゃく”, ”びゃく”, “せん”.
  “まん”, “おく”, … }
- 次の問題を考えよ。
  日本語で、“ぜろ” は必要か？
  “ひゃく” と “びゃく” は、どう使い分けられるか？

2のn乗

二つのものを重複を許してn個並べる順列の数

2のn乗
二つのものを重複を許してn個並べる順列の数

- 2のn乗は、二つのものを重複を許してn個並べる順列の数である。それは、二進数でn桁の0と1の並びの数を考えればわかる。ただし、n桁に満たない二進数は、先頭に0を補って n桁にする。

- n ＝ 3 の場合は、次の2^3=8通り。
  000, 001, 010, 011, 100, 101, 110, 111

- n ＝ 4 の場合は、次の2^4=16通り。
  0000,0001,0010,0011,0100,0101,0110,0111
  1000,1001,1010,1011,1100,1101,1110,1111

pのn乗
語の数と文の数

- 同様に、pのn乗は、p個のものを重複を許してn個並べる順列の数である。

- 26文字のアルファベット15文字以内で構成される語の数は、高々、26^{15}である。
- ただし、語彙が10万個ある言語での10個の語からなる文の数は、$100000^{10}=10^{50}$種類もある!

この$_1$ 文$_2$ は$_3$ 10$_4$ 個$_5$ の$_6$ 語$_7$ から$_8$ できて$_9$ いる$_{10}$

- 10語文というのは、そんなに長い文章ではないが、10^{50}というのは、とても巨大な数である。

語と文の複雑さの違いについて

- 先には、語彙の数を10万として計算したが、現実には語彙の数は、もっと多い。日本語の辞書の収録語彙数をあげておく。
  - 『日本国語大辞典』(小学館)　50万語
  - 『広辞苑』(第六版、岩波書店)　約24万語
  - 『岩波国語辞典』(第七版)　6万5000語
- それでも、語の数は有限である。それに対して、文の数は、可能的には無限である。辞書は存在するが、すべての文を網羅した用例集は、存在しえない。ただし、可能な語のすべての組み合わせを考える必要はない。「文法」が、その構造を与えている。

文字をコード化する
2^n への還元

- 語を構成する文字と、文を構成する語には、複雑さの階層の違いは存在するのだが、文も文字から構成される。
- p個の文字からアルファベットが構成される言語の、n文字からなる文は、p^n 個存在しうる。
- もし、p個のアルファベットのそれぞれが、最大mの長さのビット列に「コード化」されるなら、可能な文の数は、高々 2^{mn}個になる。
- こうして、可能な文の数は、高々2^Nの形で表現できる。

2のn乗
n個の要素を持つ集合の、部分集合の数

n個の要素を持つ集合の、部分集合の数を考える

- 二個の要素 a,bを持つ集合S_2の部分集合 s_iを枚挙してみよう。
  S_2 = { a, b }
  s_0 = {} , s_1={a}, s_2={b}, s_3={a,b}
- 三個の要素 a,b,cを持つ集合S_3の部分集合 s_iを枚挙してみよう。S_3 = { a, b, c }
  s_0 = {}, s_1={a}, s_2={b}, s_3={a,b},
  s_4 = {c}, s_5={a,c}, s_6={b,c}, s_7={a,b,c}
- 四個の要素 a,b,c,dを持つ集合S_4の部分集合 s_iを枚挙してみよう。S_4 = { a, b, c, d }
  s_0 = {}, s_1={a}, s_2={b}, s_3={a,b},
  s_4 = {c}, s_5={a,c}, s_6={b,c}, s_7={a,b,c},
  s_8 = {d}, s_9={a,d}, s_{10}={b,d}, s_{11}={a,b,d},
  s_{12} = {c,d}, s_{13}={a,c,d}, s_{14}={b,c,d}, s_{15}={a,b,c,d}

n個の要素を持つ集合の、部分集合の数は2^nである

帰納法で証明する。

- k=1の時、S_1={a}の部分集合は、{}と{a}だから、その数は、$2=2^1$。
- k=nの時、定理が成り立つと仮定する。
  n個の要素を持つ集合S_nに対してS_nに含まれない要素 x をとって $S_{n+1} = S_n \cup \{x\}$ とする。S_{n+1}の要素の数はn+1である。
  S_{n+1}の部分集合は、xを含むものとxを含まないものからなる。
  S_{n+1}の部分集合で、xを含まないものはS_nの部分集合に等しい。その数は、帰納法の仮定から2^nである。
  S_{n+1}の部分集合で、xを含むものは、S_nの部分集合をs_iとした時、$s_i \cup \{x\}$ の形をしているから、その数は、S_nの部分集合の数に等しい。その数は、2^nである。
  よって、S_{n+1}の部分集合の数は、$2^n + 2^n$で、2^{n+1}である。

n個の要素を持つ集合の、部分集合の数は2^nである

- n個の要素を持つ集合S_nの要素をe_iとする。
  $S_n = \{ e_0, e_1, e_2, \cdots, e_{n-1} \}$
- この時、0か1の値を取るb_iを要素とする集合C_n^xを考える。
  $C_n^x = \{ b_0, b_1, b_2, \cdots, b_{n-1} \}$
- b_iが0の時、対応するe_iを要素として含まず、
  b_iが1の時、対応するe_iを要素として含む、集合s_xを考える。
  s_xは、S_nの部分集合となる。
  また、S_nの全ての部分集合は、あるC_n^xで特徴付けられる。
- 例えば、S_3={a,b,c}の時、 C_3^0 ={0,0,0}はs_0={}に対応し、
  C_3^5={1,0,1}はs_5={a,c}に対応する。
- よって、S_nの部分集合の数は、C_n^xの数に等しい。
  それは、0と1のn個の順列の数であるので、2^nに等しい。

2のn乗は関数を表す

自然数を、
自分より小さい全ての自然数の集合と考える。

0 = Φ （ Φ は「空集合」。0より小さい自然数は存在しない。）

1 = { 0 }

2 = { 0, 1 }

3 = { 0, 1, 2 }

4 = { 0, 1, 2, 3 }

… … …

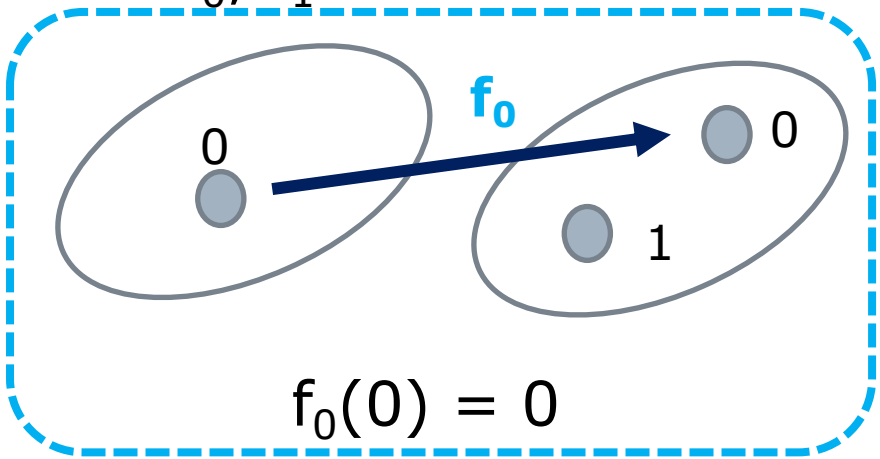
n = { 0, 1, 2, … , n−1 }

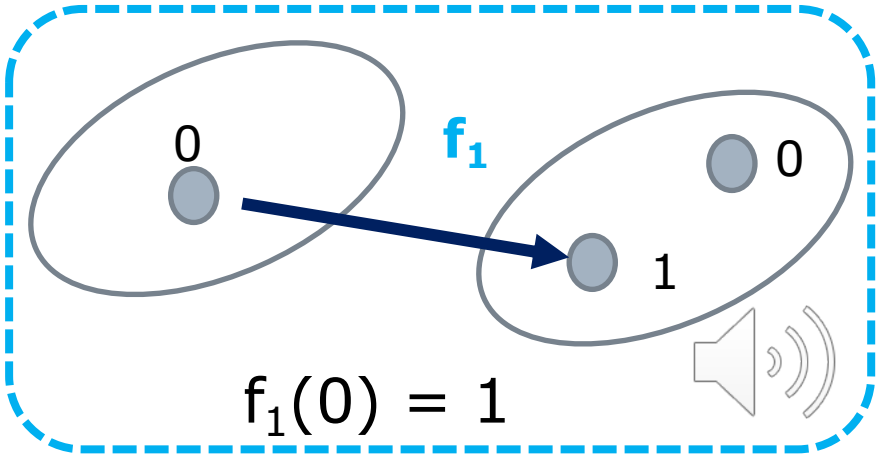
2のn乗は関数を表す?

この時、2^n を n個の要素を持つ n = { 0, 1, 2, … , n−1 } から 2個の要素を持つ 2 = { 0, 1 } への関数の集まりを表すと考えることができる。

一番簡単な n=1 の場合を考えてみよう。

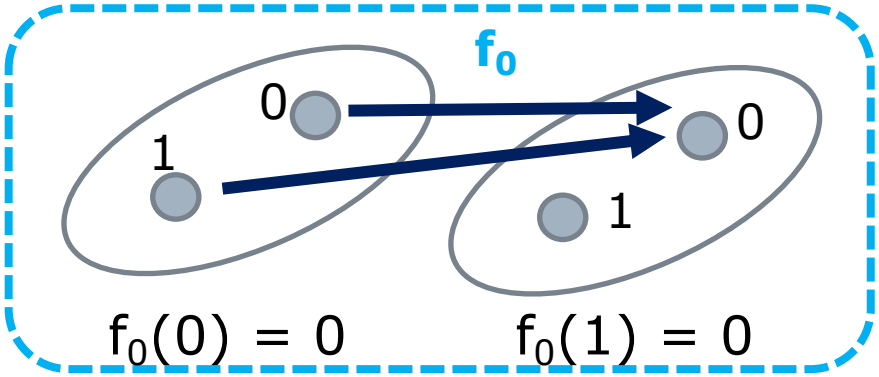
2^1には、1 = { 0 } から 2 = { 0, 1 }への次のような、2つの関数 f_0, f_1 が属する。

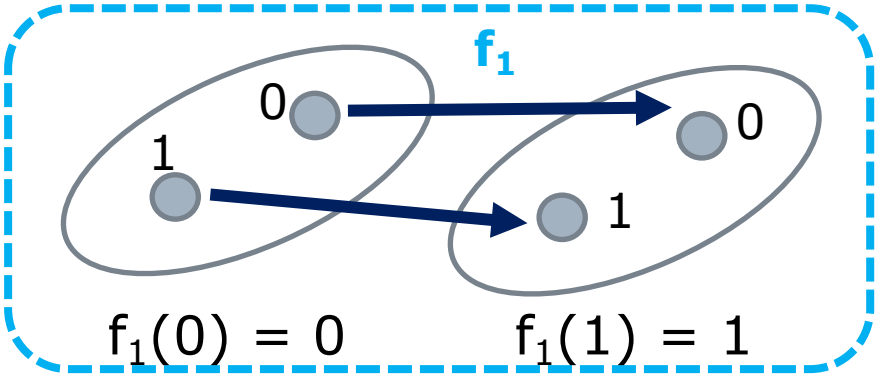


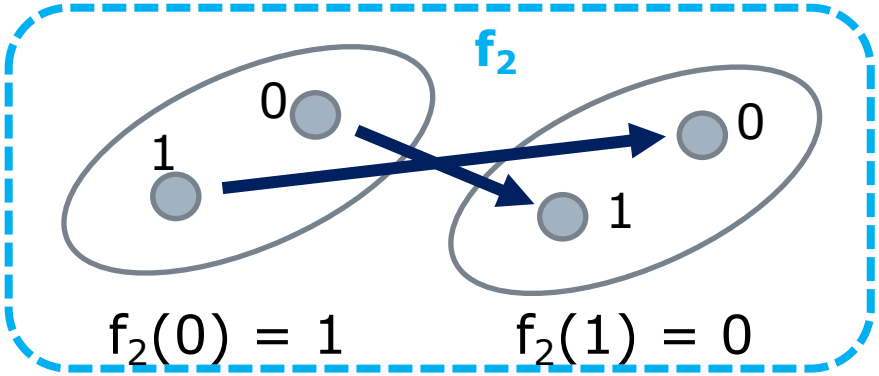


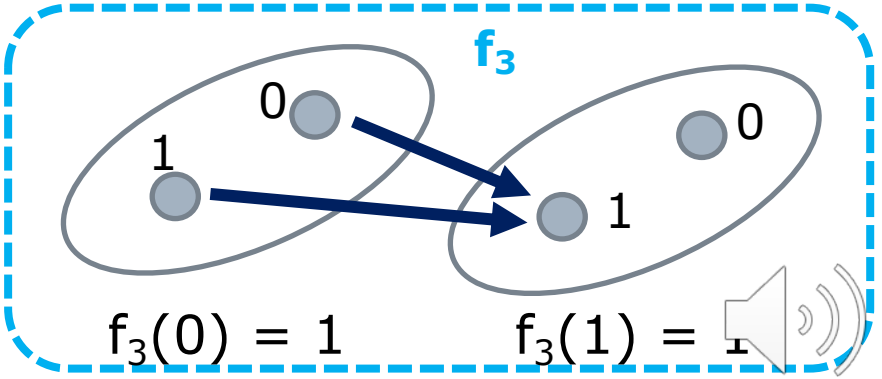
2のn乗は関数を表す?
n=2 の場合

2^2には、2 = { 0, 1 } から 2 = { 0, 1 }への次のような、4つの関数 f_0, f_1, f_2, f_3が属する。



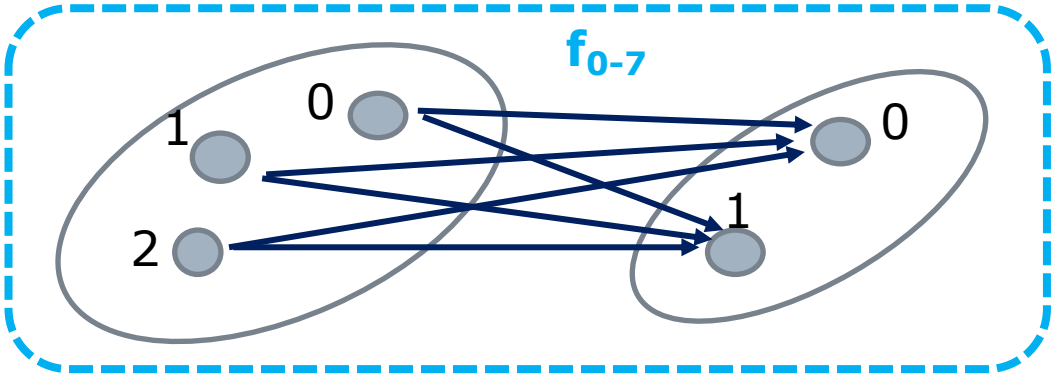






2のn乗は関数を表す?
n=3 の場合

2^3には、3 = { 0, 1, 2 } から 2 = { 0, 1 }への次のような、8つの関数 $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7$ が属する。



f_0: $f_0(0)=0, f_0(1)=0, f_0(2)=0$

f_1: $f_1(0)=0, f_1(1)=0, f_1(2)=1$

f_2: $f_2(0)=0, f_2(1)=1, f_2(2)=0$

f_3: $f_3(0)=0, f_3(1)=1, f_3(2)=1$

f_4: $f_4(0)=1, f_4(1)=0, f_4(2)=0$

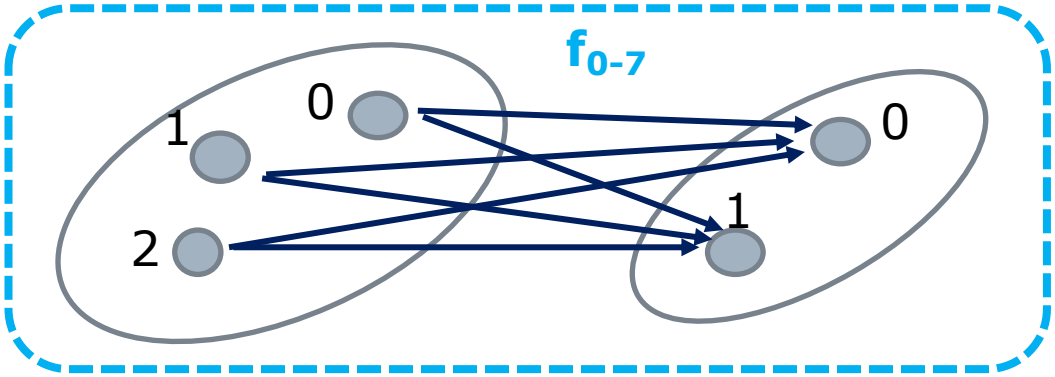
f_5: $f_5(0)=1, f_5(1)=0, f_5(2)=1$

f_6: $f_6(0)=1, f_6(1)=1, f_6(2)=0$

f_7: $f_7(0)=0, f_7(1)=0, f_7(2)=1$

2のn乗は関数を表す?
n=3 の場合(別表記)

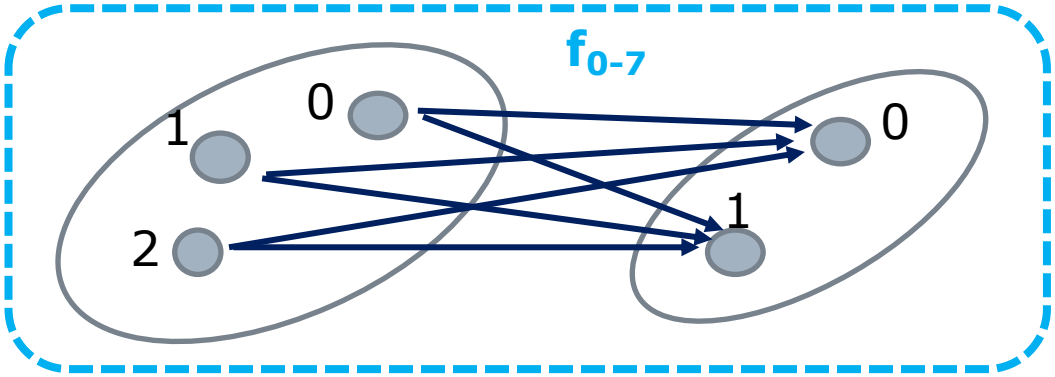
2^3には、3 = { 0, 1, 2 } から 2 = { 0, 1 }への次のような、
8つの関数 f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7 が属する。



f_0:	0,	0,	0	f_1:	0,	0,	1
f_2:	0,	1,	0	f_3:	0,	1,	1
f_4:	1,	0,	0	f_5:	1,	0,	1
f_6:	1,	1,	0	f_7:	1,	1,	1

2のn乗は関数を表す?
n=3 の場合(別表記)

2^3には、3 = { 0, 1, 2 } から 2 = { 0, 1 }への次のような、8つの関数 $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7$ が属する。



f_0:	000	f_1:	001
f_2:	010	f_3:	011
f_4:	100	f_5:	101
f_6:	110	f_7:	111

2のn乗は関数を表す?
n=3 の場合(別表記)

2^3には、3 = { 0, 1, 2 } から 2 = { 0, 1 }への次のような、8つの関数 f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7 が属する。

f_0:	000	f_1:	001
f_2:	010	f_3:	011
f_4:	100	f_5:	101
f_6:	110	f_7:	111

2のn乗は関数を表す?
n=4 の場合

2^4には、4 = { 0, 1, 2, 3 } から 2 = { 0, 1 }への次のような、16個の関数が属する。

f_0:	0000	f_1:	0001
f_2:	0010	f_3:	0011
f_4:	0100	f_5:	0101
f_6:	0110	f_7:	0111
f_8:	1000	f_9:	1001
f_{10}:	1010	f_{11}:	1011
f_{12}:	1100	f_{13}:	1101
f_{14}:	1110	f_{15}:	1111

2のn乗は関数を表す

2^n は、n = { 0, 1, 2, … , n-1 } から 2 = { 0, 1 }への

2^n 個の関数を表す。

b_j ($0 \leq j < n$)は、0 または 1の値を取るものとすると、

2^n に属する関数 f_i ($0 \leq i < 2^n$)は次のように、n個の0 または 1 の値の並びで 表現される。

$$2^n \ni f_i : \ b_0 b_1 b_2 b_3 \ \ldots b_{n-1}$$

ただし、$f_i(0) = b_0$, $f_i(1) = b_1$, $f_i(2) = b_2$,…, $f_i(k) = b_k$ …

また、i は、二進数としてみた $b_0 b_1 b_2 b_3 \ \ldots b_{n-1}$ の値に等しい。

2^nで、nを無限にまで増やしてみる

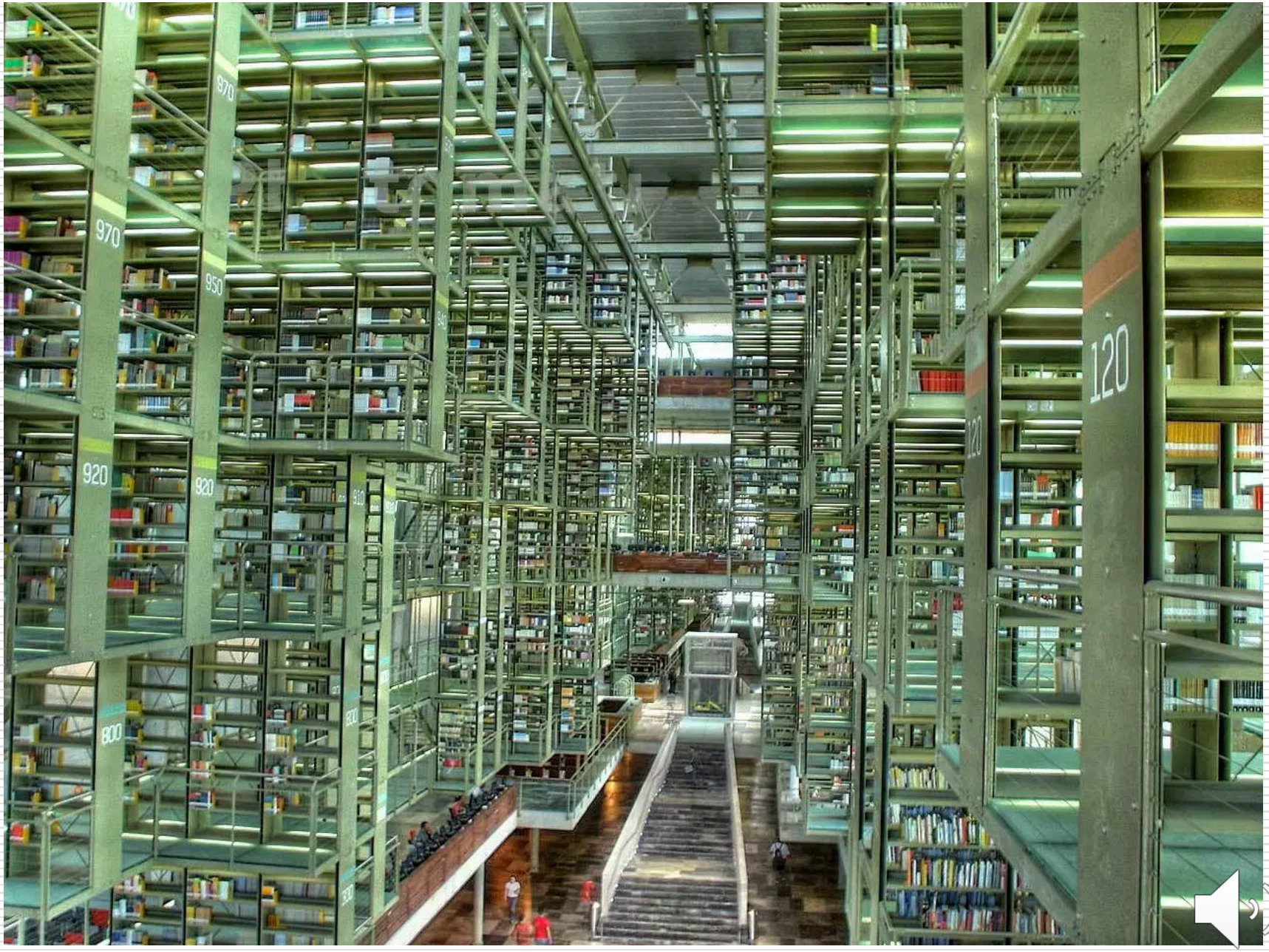
ωを、すべての自然数の集まりとする。

$\omega = \{ 0, 1, 2, 3, \ldots, n, \ldots \}$

この時、2^ωを考える。

2^ωは、次のようなものである。

- 自然数すべての上で定義された { 0, 1 }に値を持つ全ての関数の集まり。
- 無限の長さの 0と1の並び


970
950
920
800
120


$\mathfrak{c} = 2^{\aleph_0}$ ：連続体仮説

Agenda

$\mathfrak{c} = 2^{\aleph_0}$: 連続体仮説

- 「個数を数える」を分析する
  - 数えることと個数
- 無限集合の要素の個数 -- カントールが考えたこと
  - 無限集合の不思議 -- 部分は、全体と同じ数の要素を持つ
  - ヒルベルトの無限ホテル
- 0と1を結ぶ直線上の点の数
  - 平面上の点の数
  - カントール集合：奇妙な図形上の点の数
  - 不思議で奇妙な例
- カントールが証明できなかったこと --「連続体仮説」
- 非カントール的集合論の発見
  - 連続体仮説と集合論の無矛盾性の証明
  - 連続体仮説と集合論の独立性の証明

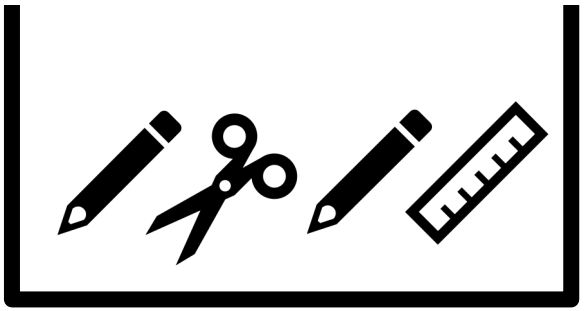
「個数を数える」を分析する

「数える」を考える

- ある箱に何かが入っているとしよう。箱に入っているものが、何個かを数えることを考える。いろいろな場合がある。
  - 同じ形の白い玉が入っている。
  - 白い玉に、赤い玉が混じっている。
  - いろんな色の同じ大きさの玉が入っている。
  - 大きさの違う、様々の色の玉が入っている。
  - 様々な形の、いろいろな色の物体が入っている。
  - 仔犬と仔猫が入っている。
  - 仔犬と仔猫と鳩が入っている。
  - 仔犬と仔猫と鳩とゴキブリが入っている。
  - ・・・・

「数える」には、いくつかの前提・特徴がある

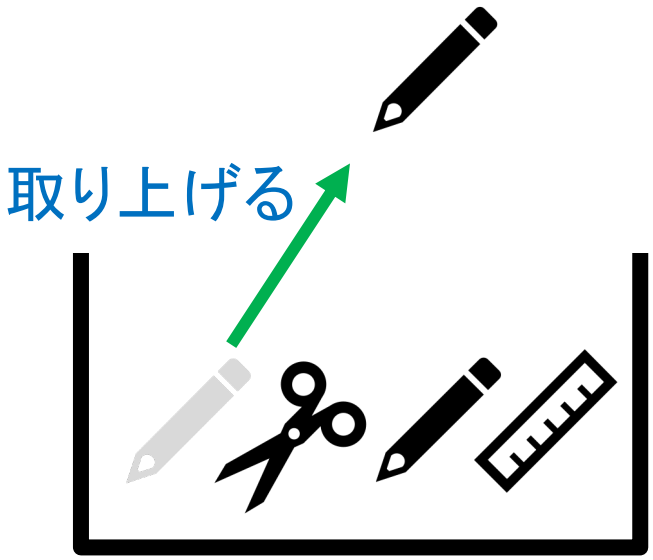
- 前提：数える範囲が明確（例えば「箱」の中にある）で、かつ、その数が、数える間に変化しないこと。
- 数えられる「もの」の具体的な形状・性質は、数えることには影響を与えない。数えるときに、それらは捨象される。
- どれを先に数えるか、同じことだが、数える順番は、数えることには影響を与えない。
- ただ、すでに数えたものと、まだ数えていないものとの区別は必要である。数えるというのは、対象をすでに数えたものとまだ数えていないものに二分することなしには実行できない。
- 最初に数えたとき1と数え、次に2と数え、順次、それを続ける。数えることは、自然数の順番に対応している。



まだ数えてないもの

すでに数えたもの

1, 2, 3, 4, 5, …

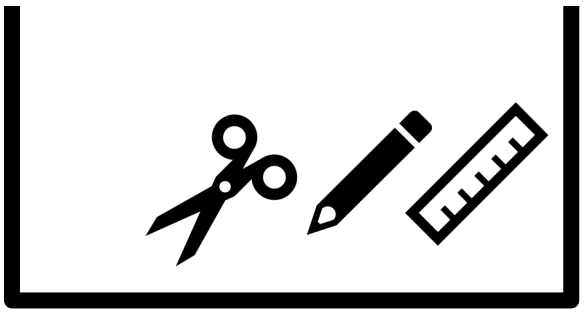


まだ数えてないもの

すでに数えたもの

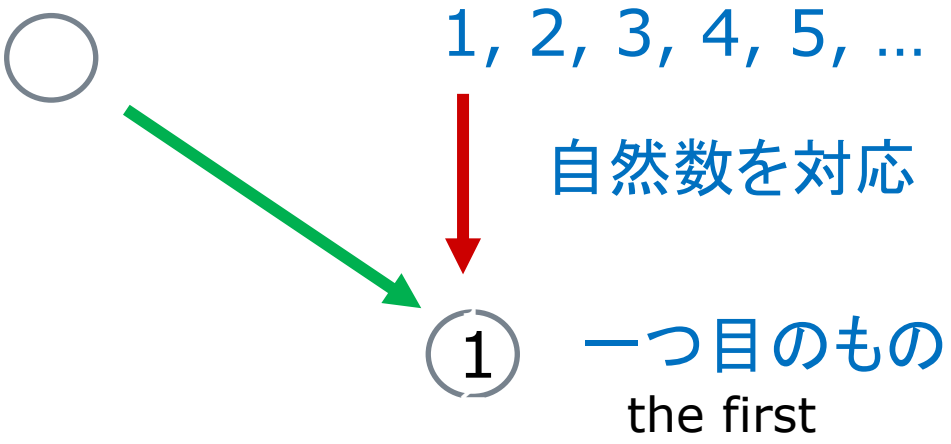
1, 2, 3, 4, 5, …

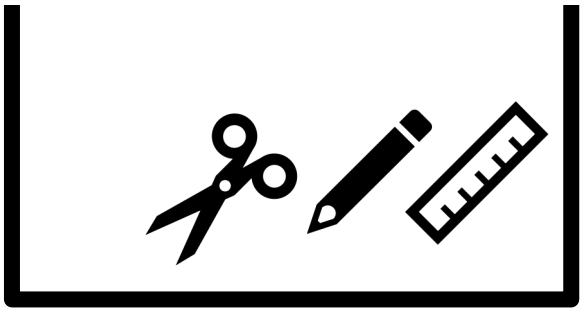
性質を捨象



まだ数えてないもの

すでに数えたもの





まだ数えてないもの

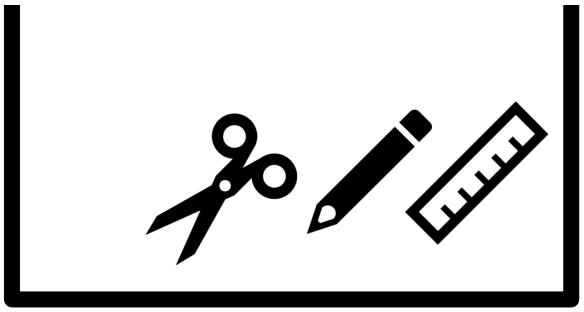
すでに数えたもの

1, 2, 3, 4, 5, …

自然数を対応

① 一つ目のもの
the first

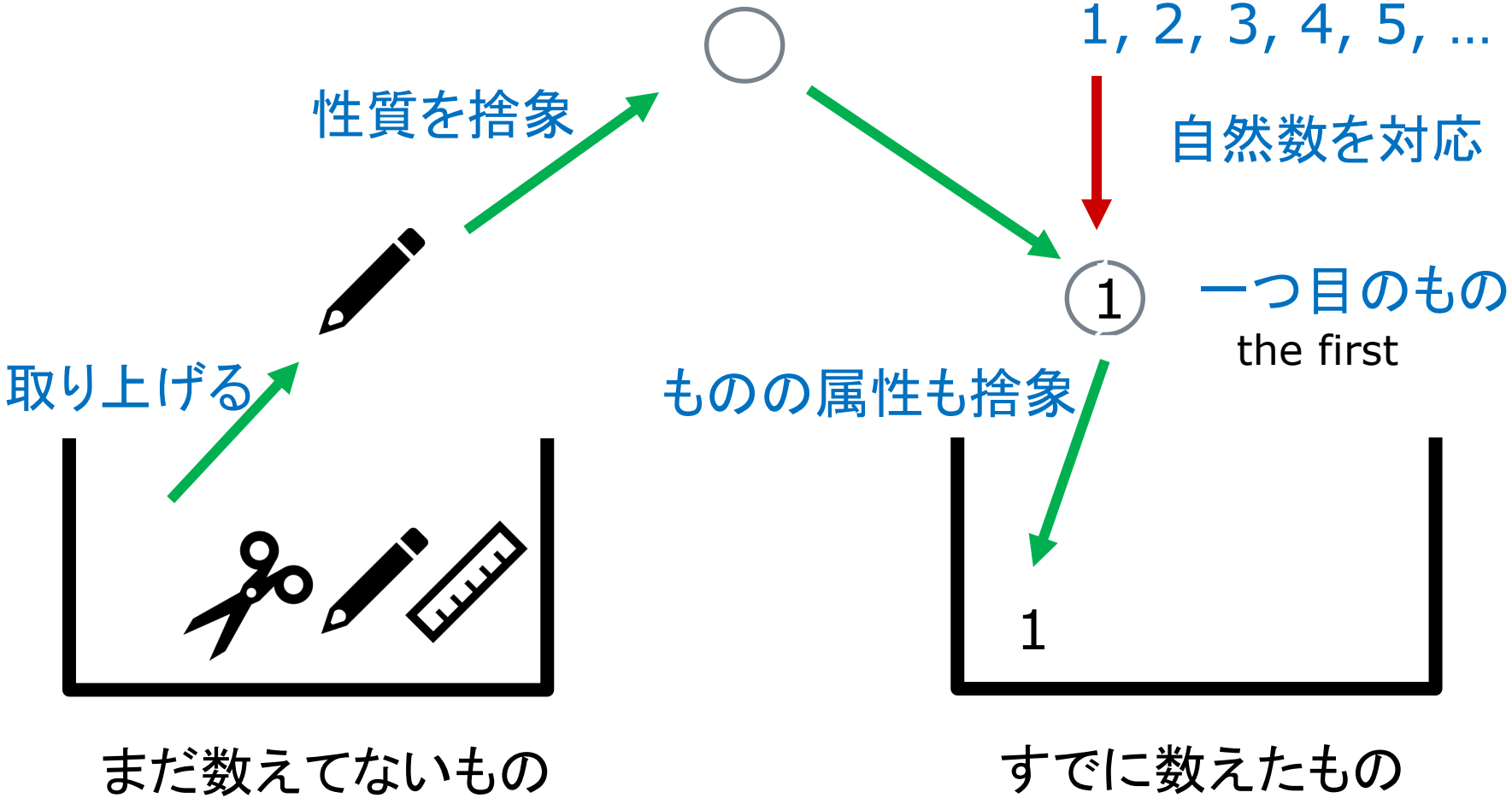
ものの属性も捨象

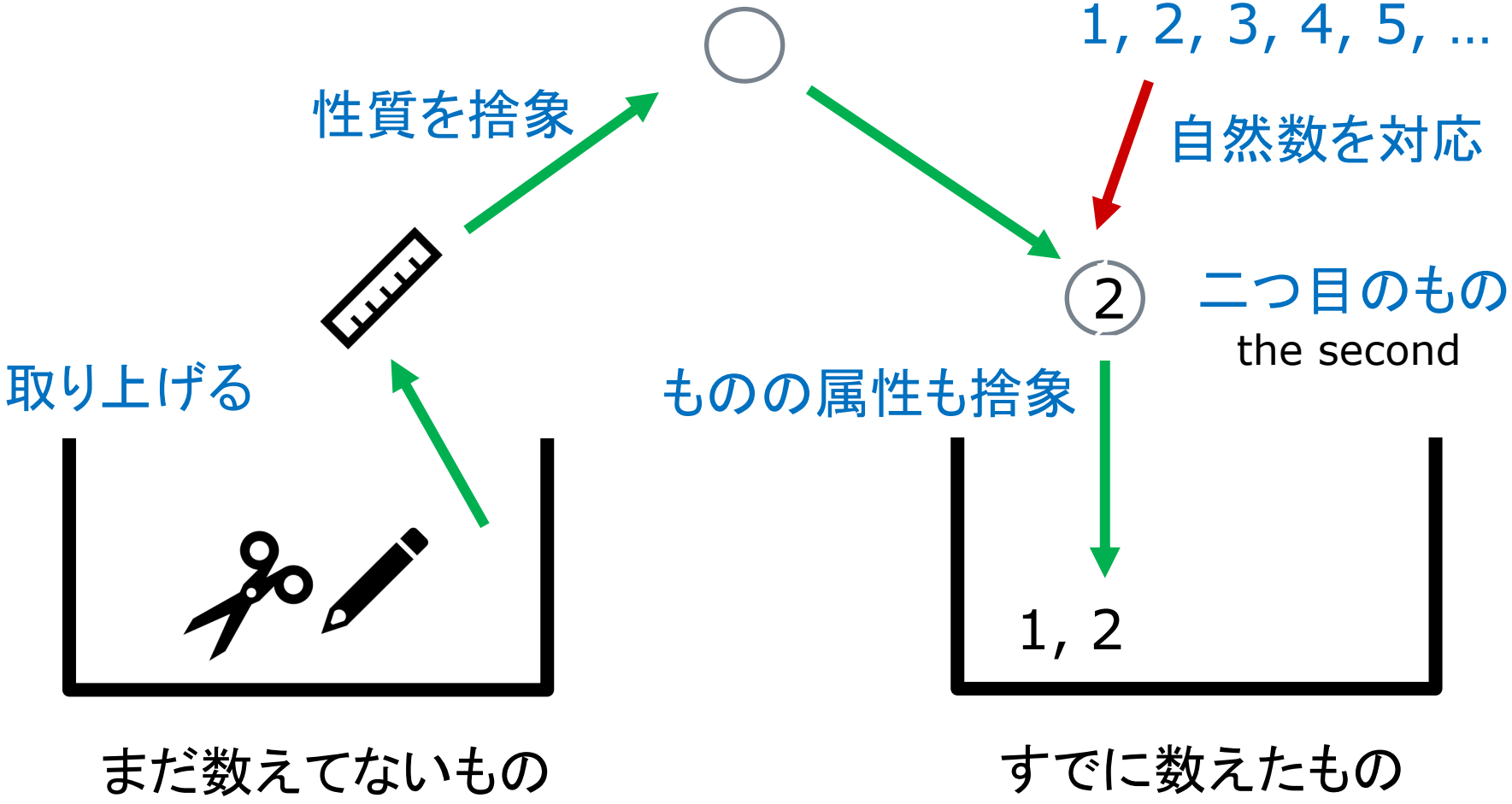


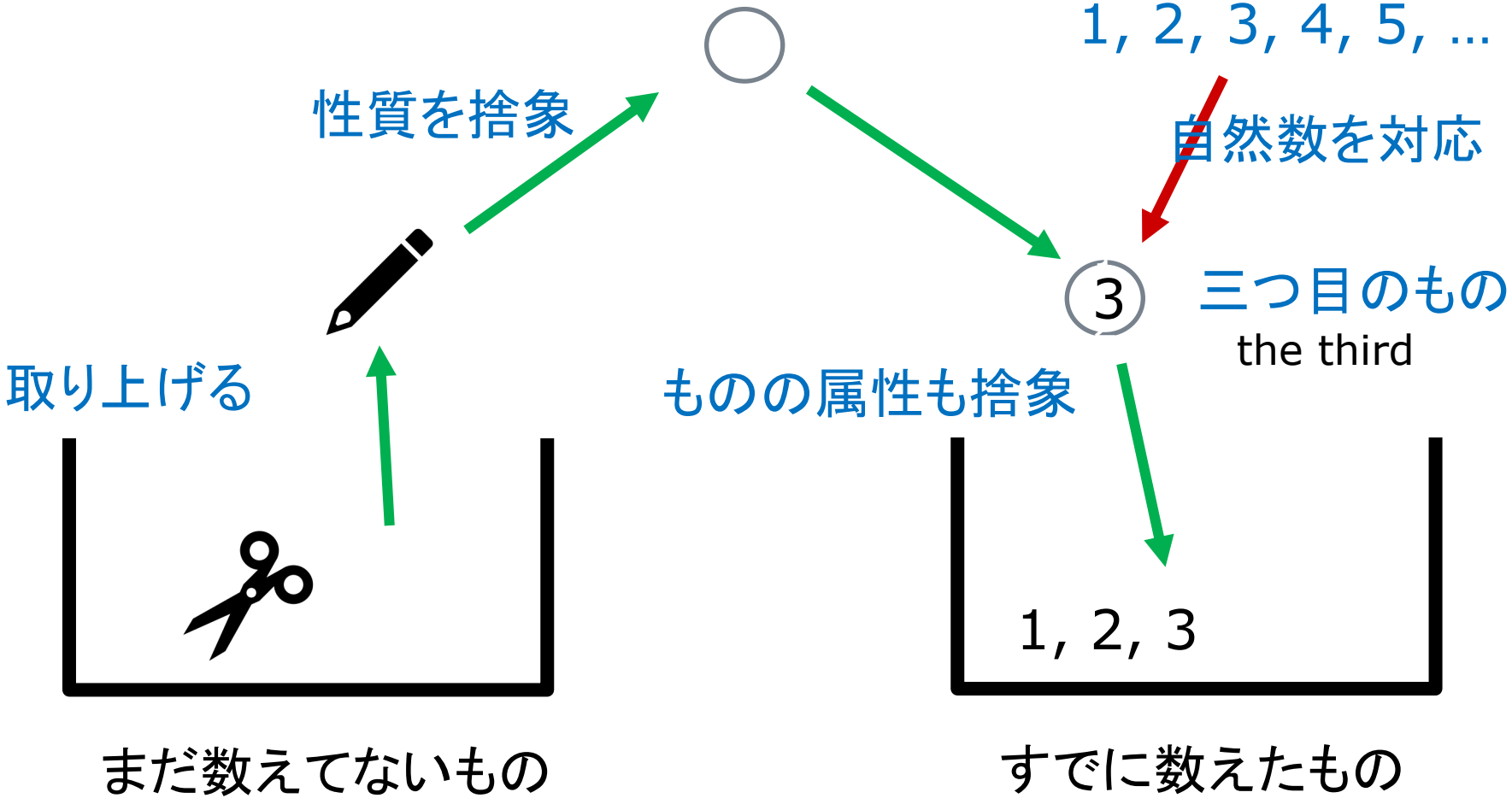
1

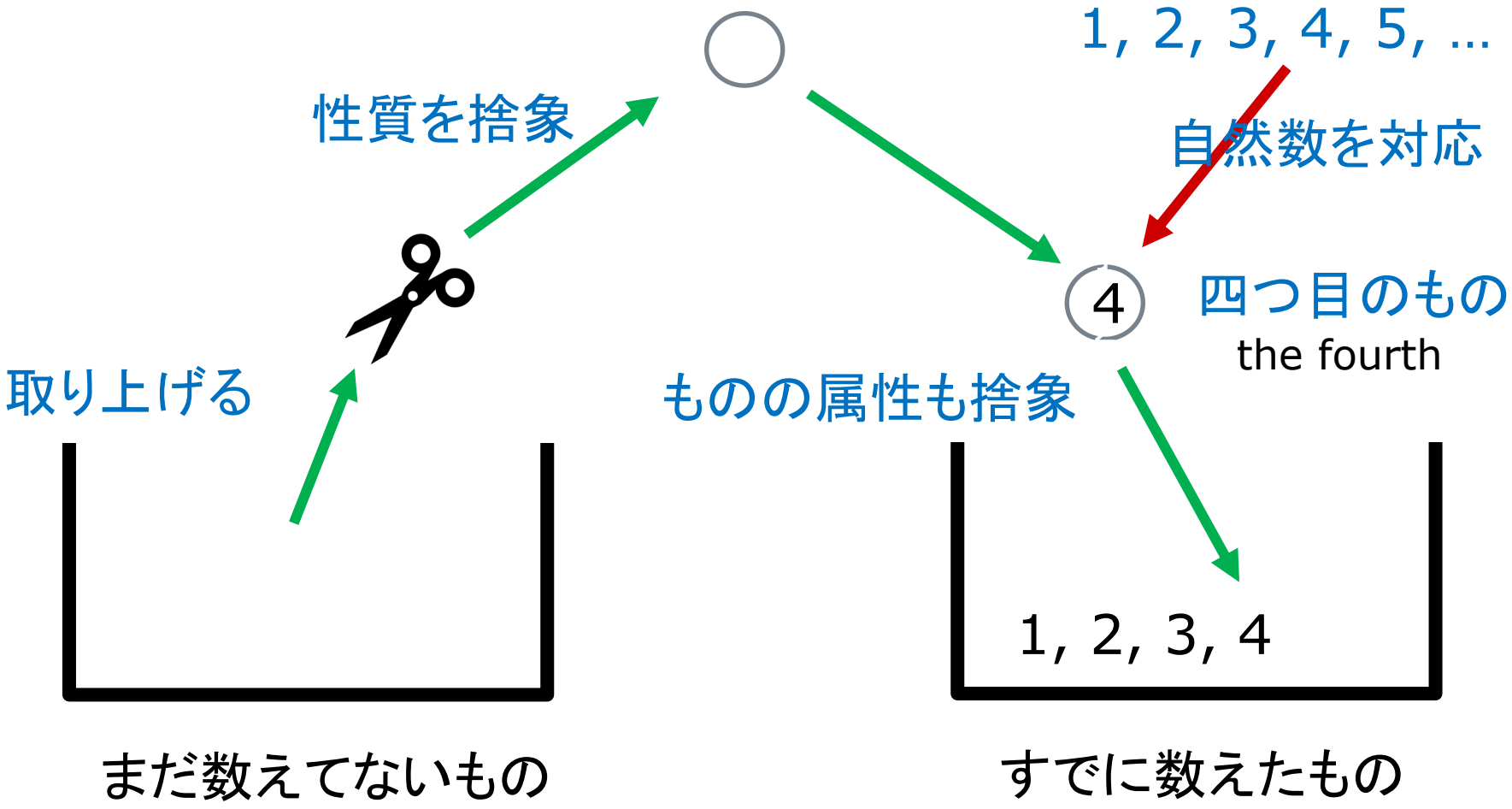
まだ数えてないもの

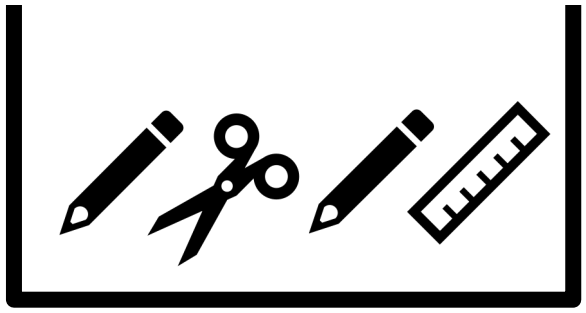
すでに数えたもの

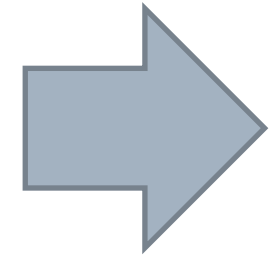

1, 2, 3, 4, 5, …
自然数を対応
性質を捨象
1
一つ目のもの
the first
取り上げる
ものの属性も捨象
1
まだ数えてないもの
すでに数えたもの

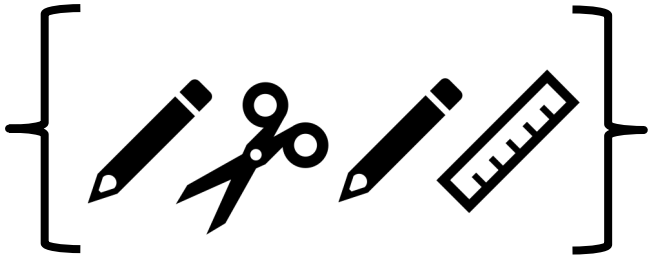

1, 2, 3, 4, 5, …
自然数を対応
性質を捨象
2
二つ目のもの
the second
取り上げる
ものの属性も捨象
1, 2
まだ数えてないもの
すでに数えたもの


性質を捨象
1, 2, 3, 4, 5, …
自然数を対応
3
三つ目のもの
the third
取り上げる
ものの属性も捨象
1, 2, 3
まだ数えてないもの
すでに数えたもの


1, 2, 3, 4, 5, …
性質を捨象
自然数を対応
4
四つ目のもの
the fourth
取り上げる
ものの属性も捨象
1, 2, 3, 4
まだ数えてないもの
すでに数えたもの







数える

{ 1, 2, 3, 4 } 順序数
Ordinals

4 個

個数

基数
Cardinals

順序の重要性　「時そば」

「おい、蕎麦屋さん。生憎と、細けえ銭っきゃ持ってねえんだ。落としちゃいけねえ、手え出してくれ」

「一（ひい）、二（ふう）、三（みい）、四（よう）、五（いつ）、六（むう）、七（なな）、八（やあ）」

「今何時（なんどき）でい!」

「へい、九（ここの）つでい」

「十（とう）、十一、十二、十三、十四、十五、十六、御馳走様」

欠落

{ 1,2,3,4,5,6,7,8} + 9 + {10,11,12,13,14,15,16}

8 + **7** = **15**

「一、二、……八、今何時でい」

「へい、四つでい」

「五、六……」

二重カウント

{ 1,2,3,4,5,6,7,8} + 4 + {5,6,7,8,9,10,11, 12,13,14,15,16}

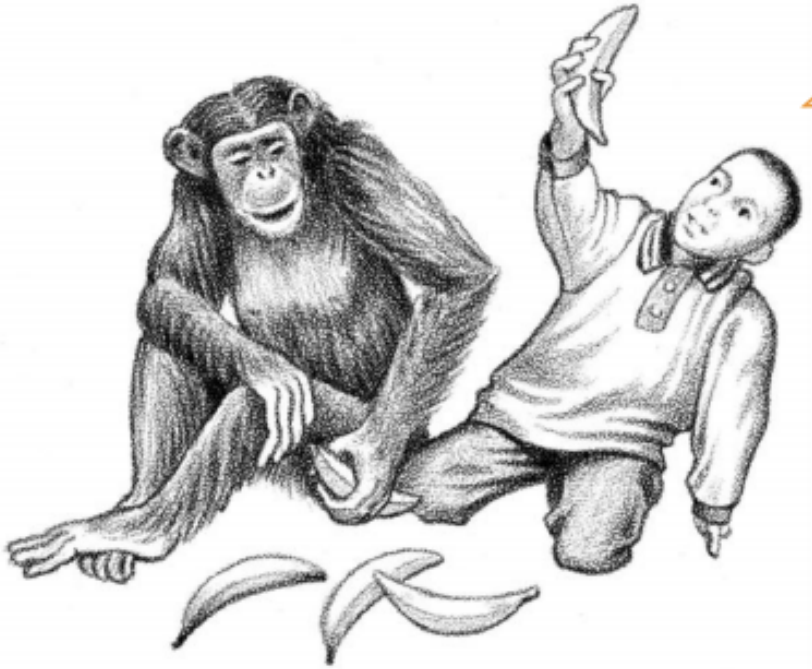
8 + **12** = **20**

The Faculty of Language: What Is It, Who Has it, and How Did It Evolve?

http://www.chomsky.info/articles/20021122.pdf

言語のRecursionの能力は、数の概念の形成能力とも結びついている。ハトは、3以上の数を区別出来ない。サルは、個別の数を個別に学ぶが、人間の子のように一般化出来ない。

5
4
3
2
1



5
4
3
2
1

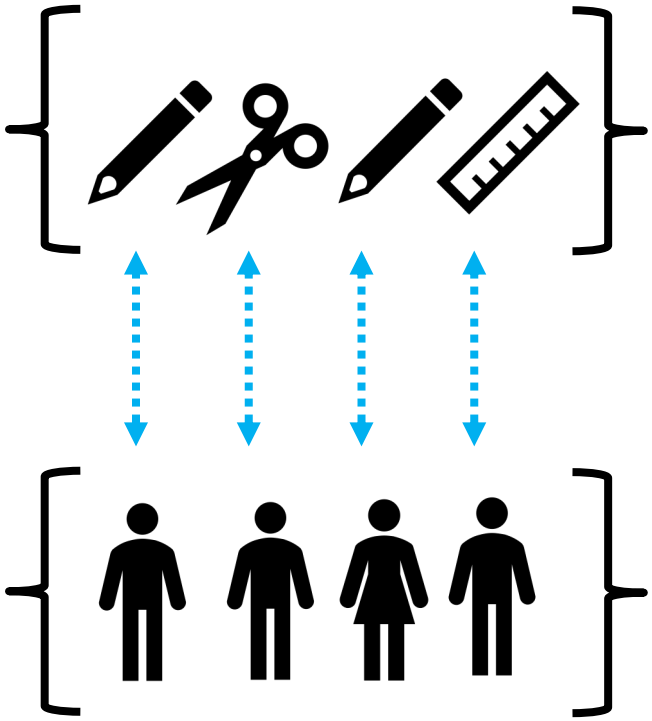
数えることと個数

「個数」について考える

- 数えることと「個数」とは、もともとは結びついていたが、そのうちに、数えることとは独立にでも、「個数」は存在するという考え方が、一般的になる。
- アルキメデスの宇宙を埋める砂の数や、日本野鳥の会のカウンティングは、実際には、数えていない。数えなくても、あるいは、数えられなくても、「個数」は存在すると思われている。
- 超ミクロな世界の量子物理学でも、様々な「量子数」の保存法則というのは、基本的なものだ。最終的には、自然数の間の関係式として物理法則が記述される可能性はある。
- ただ、そういう不変性が根本原理かどうかは、僕には、わからない。

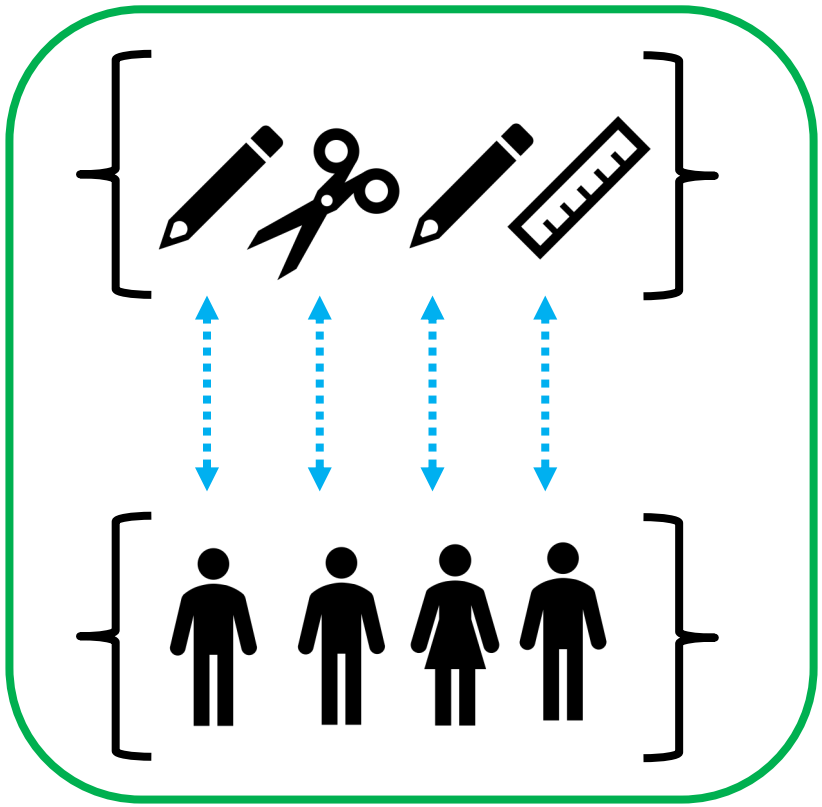
「数える」とは独立な「個数」の定義

二つの集合の要素が、「一対一対応」がつく時、
二つの集合の要素の個数は、等しいとする。



「数える」とは独立な「個数」の定義

二つの集合の要素が、「一対一対応」がつく時、
二つの集合の要素の個数は、等しいとする。



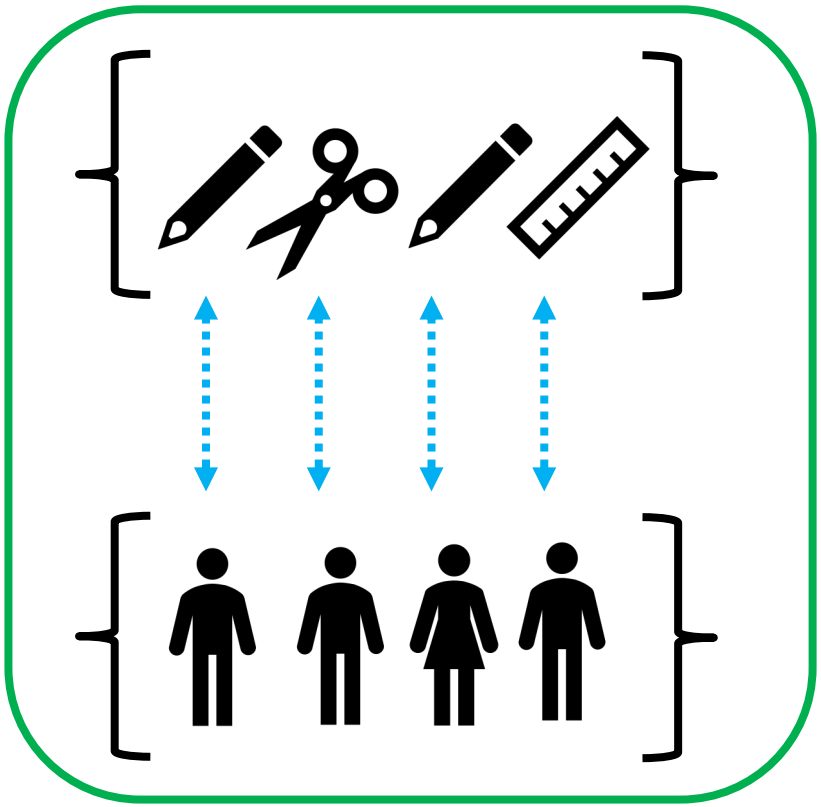
集合の要素の数が有限の時、両者の定義は一致する。ただし、問題は無限集合の要素の数を考えた時に起きる。

{ 1, 2, 3, 4 }

個数 4

先に次のような、「数える」とは独立な「個数」の定義にあげた。無限集合の場合はどうなるか？

二つの集合の要素が、「一対一対応」がつく時、
二つの集合の要素の個数は、等しいとする。



ただし、集合の要素の数が有限の時、両者の定義は一致する。問題は無限集合の要素の数を考えた時に起きる。

{ 1, 2, 3, 4 }

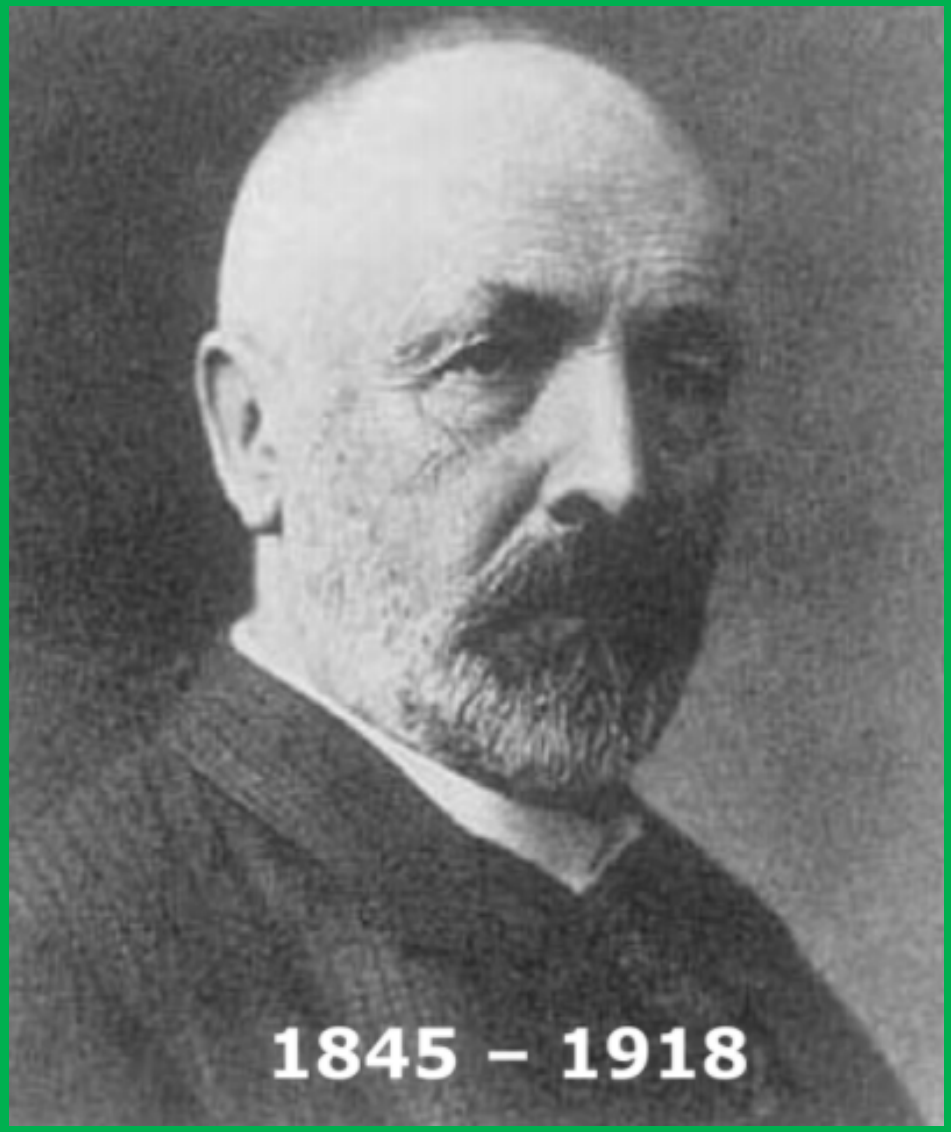
個数 4

無限集合の要素の個数

カントールが考えたこと

「数える」こととは切り離して「個数」が存在すると書いた。

無限の順序数は、たかだか「可算」なものの数え上げにしか使えない。数えられない（非可算な）ものを含めた個数の定義に、カントールは、「一対一対応」を採用する。そこで見えて来た世界は、直感を超えたものだった。


1845 – 1918

無限集合の不思議

部分は、全体と同じ数の要素を持つ

ユークリッド「幾何学原論」
公理8 「全体は、部分より大きい」

無限集合の不思議
部分は、全体と同じ数の要素を持つ

奇数・偶数は、自然数の一部である。明らかに、自然数の数は、奇数や偶数の数より大きいように見える。

自然数：　0, 1, 2, 3, 4, 5, 6, 7, 8, 9, …
奇　数：　　1,　 3,　 5,　 7,　 9, …
偶　数：　0,　 2,　 4,　 6,　 8,　…

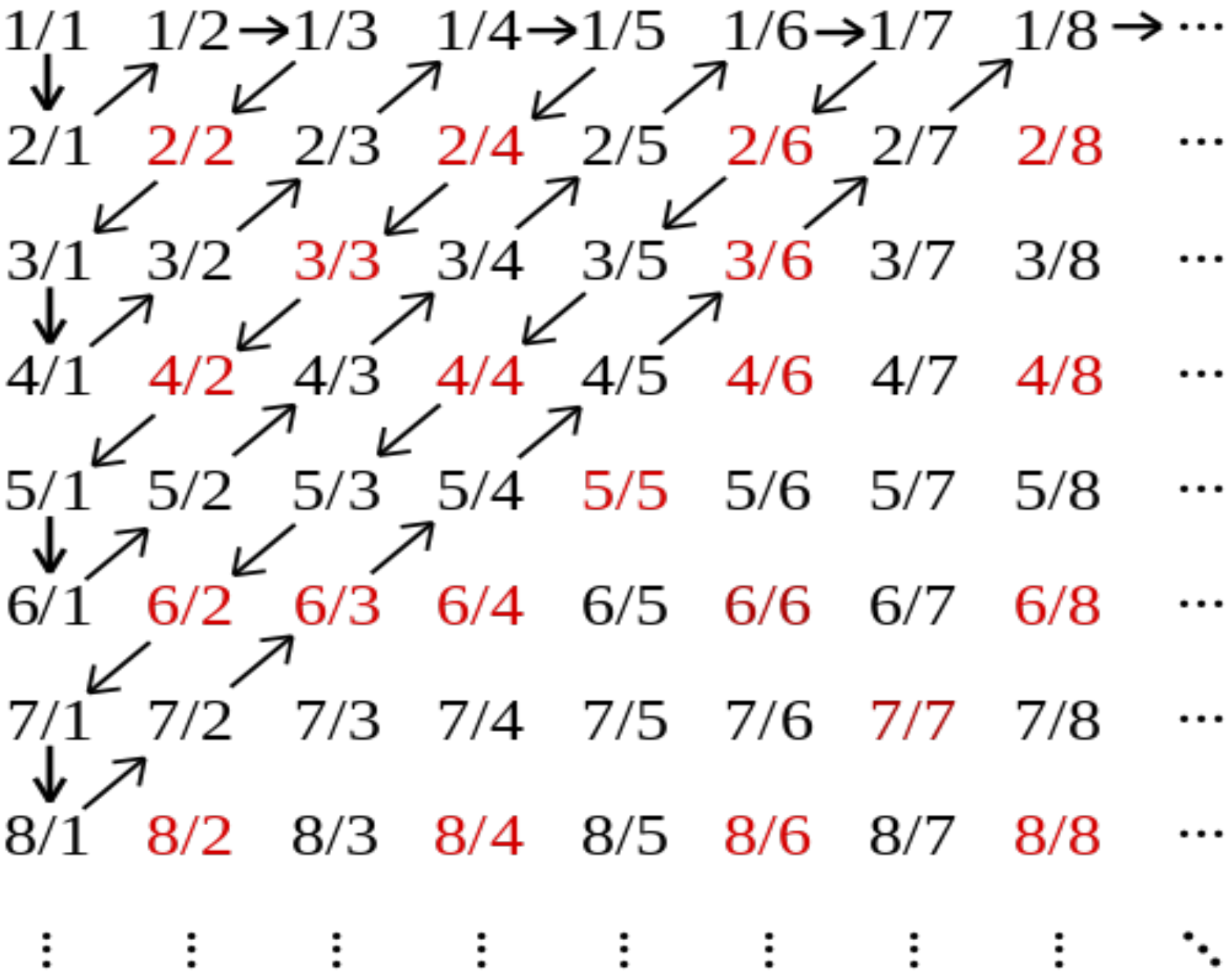
ただ、次のような対応で、奇数全体も偶数全体も、自然数全体と「一対一」に対応することがわかる。

自然数：　0, 1, 2, 3, 4, 5,　6, 7,　8, 9, …,n, …
奇　数：　1, 3, 5, 7, 9,11,13,15,17,19…,2n+1, …
偶　数：　0, 2, 4, 6, 8,10,12,14,16,18…,2n, …

よって、「一対一」対応で、要素の数を考えれば、自然数の数と奇数の数と偶数の数は等しい。

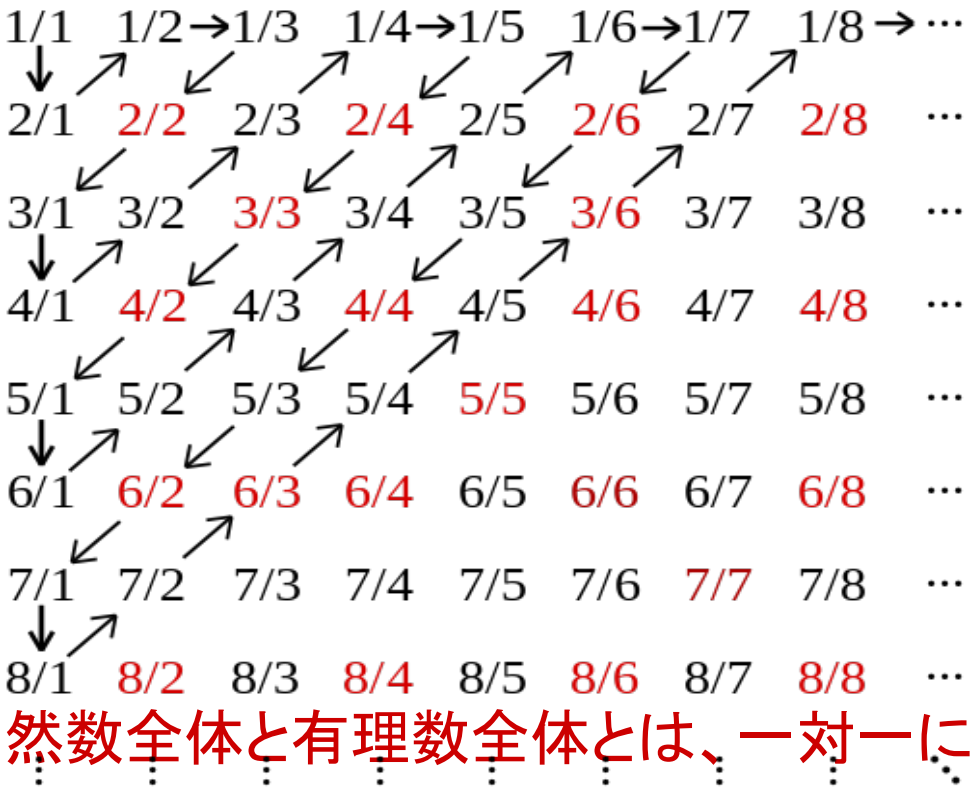
自然数全体と有理数全体とは、一対一に対応する

- 有理数は、0以外の二つの自然数p,qについて、p/q で表される数である。0と1の間にも、有理数は無限に存在する。
- ところが、次のように有理数を全て数え上げることができる。


1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 ⋯
2/1 2/2 2/3 2/4 2/5 2/6 2/7 2/8 ⋯
3/1 3/2 3/3 3/4 3/5 3/6 3/7 3/8 ⋯
4/1 4/2 4/3 4/4 4/5 4/6 4/7 4/8 ⋯
5/1 5/2 5/3 5/4 5/5 5/6 5/7 5/8 ⋯
6/1 6/2 6/3 6/4 6/5 6/6 6/7 6/8 ⋯
7/1 7/2 7/3 7/4 7/5 7/6 7/7 7/8 ⋯
8/1 8/2 8/3 8/4 8/5 8/6 8/7 8/8 ⋯
⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋱

自然数全体と有理数全体とは、一対一に対応する

- 有理数は、0以外の二つの自然数p,qについて、p/q で表される数である。0と1の間にも、有理数は無限に存在する。
- ところが、次のように有理数を全て数え上げることができる。



すなわち、自然数全体と有理数全体とは、一対一に対応する

n個の自然数の組の全体と、自然数は、一対一対応する

- 有理数は、ゼロでない二個の自然数の組と考えることができるが、ゼロでない自然数のn個の組も、次のように一つの自然数に対応づけることができる。
- 以下で、i番目の素数をp_iで表すとする。

  $p_1=2, p_2=3, p_3=5, p_4=7, p_5=11, p_6=13, \ldots$

  この時、自然数のk個の組 $(n_1, n_2, n_3, .., n_k)$に、次の数Aを対応づける。

  $A = p_1^{n_1}\ p_2^{n_2}\ p_3^{n_3} \ldots p_k^{n_k}$

  たとえば、4個の自然数の組（23, 100, 1, 13）は、次の自然数に対応づけられる。

  $(23, 100, 1, 13) \rightarrow 2^{23} \cdot 3^{100} \cdot 5^{1} \cdot 7^{13}$
- この対応は、一対一である。

  $2^{23} \cdot 3^{100} \cdot 5^{1} \cdot 7^{13} \rightarrow (23, 100, 1, 13)$

無限の不思議

ヒルベルトの無限ホテル

1862年1月23日- 1943年2月14日

無限の不思議
ヒルベルトの無限ホテル

「客室が無限にあるホテルを考える。現実にある客室が有限のホテルの場合には、「満室である」ということと「もう1人も泊められない」ということは同値である。しかし「無限ホテル」ではそうはならない。無限ホテルが「満室である」としよう。この場合でも次のようにして新たな客を泊めることができる。客室数は無限とはいえ 1, 2, 3, … と番号を付けられる。客が1人来たら、1号室にいた客を2号室へ、2号室の客を3号室へ、3号室の客を4号室へ、…、n 号室の客を $n + 1$ 号室へ、…と順番に移す。客室は無限にあるのだから誰もあぶれることはない。新たな客は1号室に泊めればよい。」

https://goo.gl/Mer4cZ

無限の不思議
ヒルベルトの無限ホテル

「新たな客は1人どころか、複数でも、（可算）無限でもよい。例えば、1号室の客を2号室へ、2号室の客を4号室へ、3号室の客を6号室へ、…、n 号室の客を $2n$ 号室へ、…と移せば、1号室、3号室、5号室、…つまり奇数号室は空室になるから、無限の客を新たに泊めることができる。」

https://goo.gl/Mer4cZ

https://goo.gl/sYdXRu

0と1を結ぶ直線上の点の数

実数の濃度は非可算であるという発見。

1874年　最初の証明

1891年　対角線論法での証明

0と1との間の実数全てを、次のように自然数と一対一に対応付けることが出来たとしよう。

$$
\begin{array}{ll}
1 & 0.a_{1,1}a_{1,2}a_{1,3}a_{1,4}a_{1,5}a_{1,6}a_{1,7}a_{1,8}a_{1,9}a_{1,10}a_{1,11}\cdots\cdots \\
2 & 0.a_{2,1}a_{2,2}a_{2,3}a_{2,4}a_{2,5}a_{2,6}a_{2,7}a_{2,8}a_{2,9}a_{2,10}a_{2,11}\cdots\cdots \\
3 & 0.a_{3,1}a_{3,2}a_{3,3}a_{3,4}a_{3,5}a_{3,6}a_{3,7}a_{3,8}a_{3,9}a_{3,10}a_{3,11}\cdots\cdots \\
4 & 0.a_{4,1}a_{4,2}a_{4,3}a_{4,4}a_{4,5}a_{4,6}a_{4,7}a_{4,8}a_{4,9}a_{4,10}a_{4,11}\cdots\cdots \\
5 & 0.a_{5,1}a_{5,2}a_{5,3}a_{5,4}a_{5,5}a_{5,6}a_{5,7}a_{5,8}a_{5,9}a_{5,10}a_{5,11}\cdots\cdots \\
6 & 0.a_{6,1}a_{6,2}a_{6,3}a_{6,4}a_{6,5}a_{6,6}a_{6,7}a_{6,8}a_{6,9}a_{6,10}a_{6,11}\cdots\cdots \\
7 & 0.a_{7,1}a_{7,2}a_{7,3}a_{7,4}a_{7,5}a_{7,6}a_{7,7}a_{7,8}a_{7,9}a_{7,10}a_{7,11}\cdots\cdots \\
8 & 0.a_{8,1}a_{8,2}a_{8,3}a_{8,4}a_{8,5}a_{8,6}a_{8,7}a_{8,8}a_{8,9}a_{8,10}a_{8,11}\cdots\cdots \\
9 & 0.a_{9,1}a_{9,2}a_{9,3}a_{9,4}a_{9,5}a_{9,6}a_{9,7}a_{9,8}a_{9,9}a_{1,10}a_{9,11}\cdots\cdots \\
 & \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots \\
 & \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots\ \ \vdots
\end{array}
$$

この自然数と実数との一対一の対応関係が存在するとしてで、対角線上の要素を選び出す。

1 $0.a_{1,1}a_{1,2}a_{1,3}a_{1,4}a_{1,5}a_{1,6}a_{1,7}a_{1,8}a_{1,9}a_{1,10}a_{1,11}\cdots\cdots$

2 $0.a_{2,1}a_{2,2}a_{2,3}a_{2,4}a_{2,5}a_{2,6}a_{2,7}a_{2,8}a_{2,9}a_{2,10}a_{2,11}\cdots\cdots$

3 $0.a_{3,1}a_{3,2}a_{3,3}a_{3,4}a_{3,5}a_{3,6}a_{3,7}a_{3,8}a_{3,9}a_{3,10}a_{3,11}\cdots\cdots$

4 $0.a_{4,1}a_{4,2}a_{4,3}a_{4,4}a_{4,5}a_{4,6}a_{4,7}a_{4,8}a_{4,9}a_{4,10}a_{4,11}\cdots\cdots$

5 $0.a_{5,1}a_{5,2}a_{5,3}a_{5,4}a_{5,5}a_{5,6}a_{5,7}a_{5,8}a_{5,9}a_{5,10}a_{5,11}\cdots\cdots$

6 $0.a_{6,1}a_{6,2}a_{6,3}a_{6,4}a_{6,5}a_{6,6}a_{6,7}a_{6,8}a_{6,9}a_{6,10}a_{6,11}\cdots\cdots$

7 $0.a_{7,1}a_{7,2}a_{7,3}a_{7,4}a_{7,5}a_{7,6}a_{7,7}a_{7,8}a_{7,9}a_{7,10}a_{7,11}\cdots\cdots$

8 $0.a_{8,1}a_{8,2}a_{8,3}a_{8,4}a_{8,5}a_{8,6}a_{8,7}a_{8,8}a_{8,9}a_{8,10}a_{8,11}\cdots\cdots$

9 $0.a_{9,1}a_{9,2}a_{9,3}a_{9,4}a_{9,5}a_{9,6}a_{9,7}a_{9,8}a_{9,9}a_{1,10}a_{9,11}\cdots\cdots$

⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮

⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮

実数は、自然数とは、「一対一」で対応しない
実数の個数は自然数の個数より大きい！

- こうしてできた対応表の対角線上の桁の数字からなる実数を、Aとしよう。

  $A = 0.a_{1,1}a_{2,2}a_{3,3}a_{4,4}a_{5,5}a_{6,6}a_{7,7}a_{8,8}a_{9,9}\ \cdots\cdots$

- Aの小数点以下の各桁の数字$a_{n,n}$を、各桁の数字に１を加えた数字$a'_{n,n}$で置き換える（9の桁は0にする）。この数字をBとする。置き換えたのだから $a'_{n,n} \neq a_{n,n}$ である。例えば、

  A = 0.341980675597 …. ならば、

  B = 0.452091786608 …. となる。

- この数字

  $B = 0.a'_{1,1}a'_{2,2}a'_{3,3}a'_{4,4}a'_{5,5}a'_{6,6}a'_{7,7}a'_{8,8}a'_{9,9}\ \cdots\cdots$

  は、先の対応表の中には存在しない。なぜなら先の対応表の中の数字とは、少なくとも一箇所では異なるから。

自然数と実数の一対一対応が存在しないことの別証明

r_i を0と1の間の実数の二進表示とする。

0と1との間の実数全てを、次のように自然数と一対一に対応付けることが出来たとしよう。

$$0 : r_0 \ , \ 1 : r_1 \ , \ 2 : r_2 \ , \ \ldots \ , \ n : r_n \ , \ \ldots$$

r_i は、$r_i \in 2^\omega$という関数としても考えることができるので、次のような 0と1の並び Dを考える。

$r_n(r_n)$ は、関数r_nに引数r_nを与えたものである。0か1が返る。

$$D = r_0(r_0)r_1(r_1)r_2(r_2) \ \ldots \ r_n(r_n) \ \ldots$$

今度は、このビット列を反転してKとする。$\overline{0}=1$, $\overline{1}=0$, $\overline{101}=010$

$$K=\overline{D}=\overline{r_0(r_0)r_1(r_1)r_2(r_2) \ \ldots \ r_n(r_n) \ \ldots}$$

自然数と実数の一対一対応が存在しないことの別証明

DもKも、0,1の無限列で、0と1の間の実数の二進表示と見なすことができる。同時に、DもKも、D, K $\in 2^{\omega}$ という関数とみなすことができる。

Kが先の対応リストに含まれているとしよう。すなわち、ある i について、$r_i = K$ としよう。この時、$r_i(r_i) = K(r_i)$ となる。

ところで、Kの定義から、$K(r_i) = \overline{r_i(r_i)}$

$r_i(r_i) = K(r_i) = \overline{r_i(r_i)}$　　これは矛盾である。

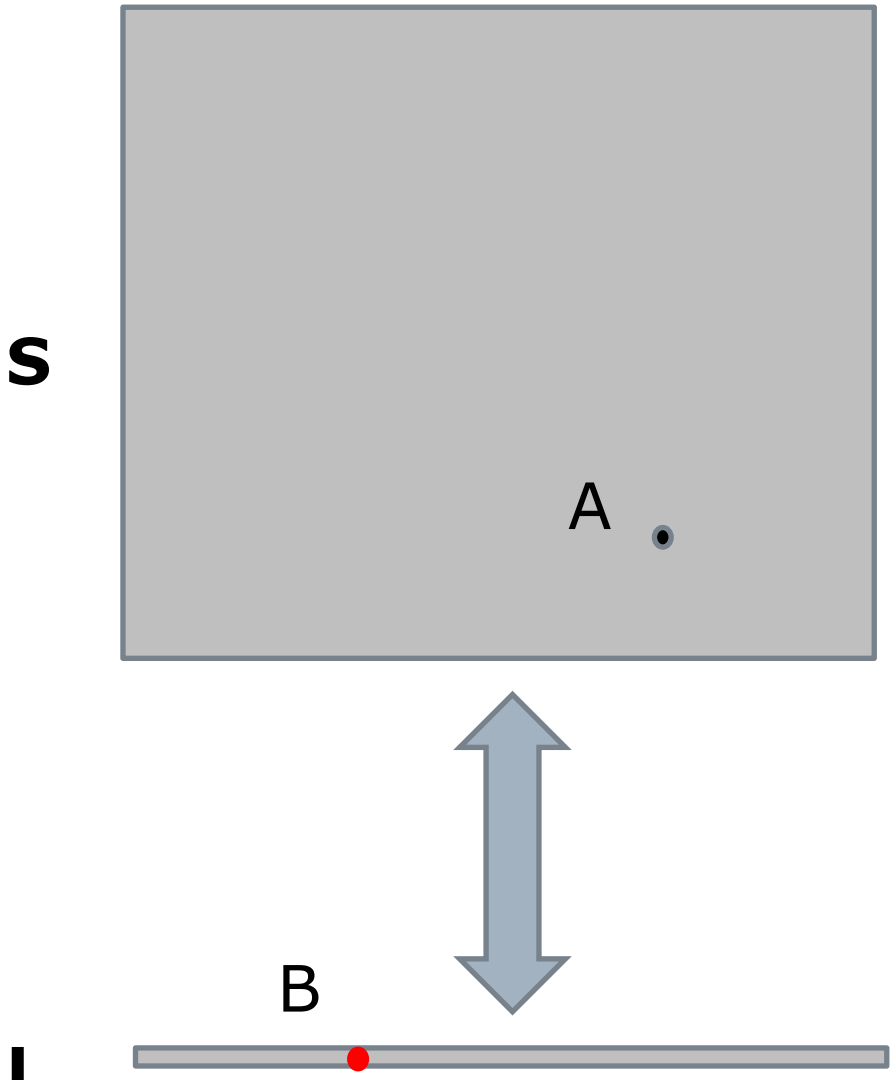
平面上の点の数

n次元のユークリッド空間R^nは、実数Rと同じ濃度を持つ。

1878年

“*Je le vois, mais je ne le crois pas*!” (“I see it, but I don‘t believe it!”) Dedekindへの手紙

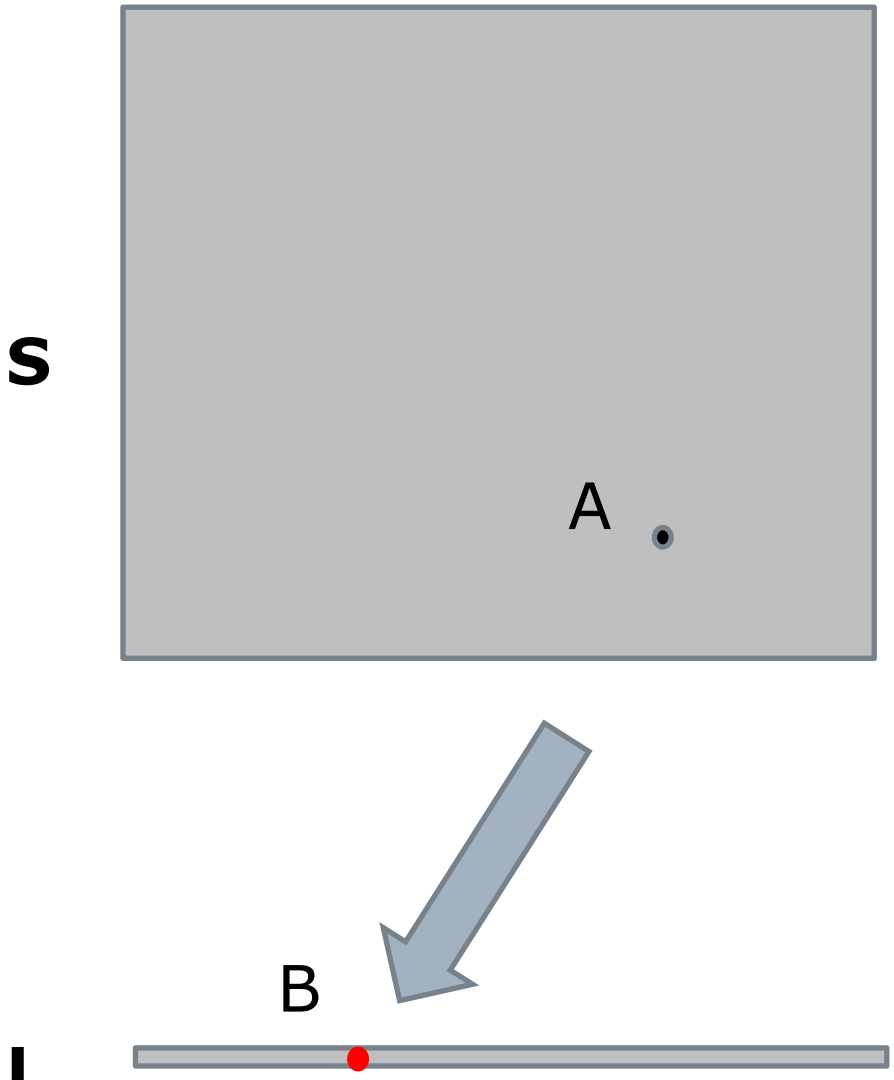
平面と直線の点の濃度



1 x 1 の大きさを持つ正方形Sと、
長さ 1 の直線Lを考えよう。

S上の点の数と、L上の点の数は
どちらが大きいだろうか？

平面と直線の点の濃度



平面上の点Aの座標を(x,y)とする。

$x=0.x_1x_2x_3x_4x_5x_6x_7x_8x_9\ldots$

$y=0.y_1y_2y_3y_4y_5y_6y_7y_8y_9\ldots$

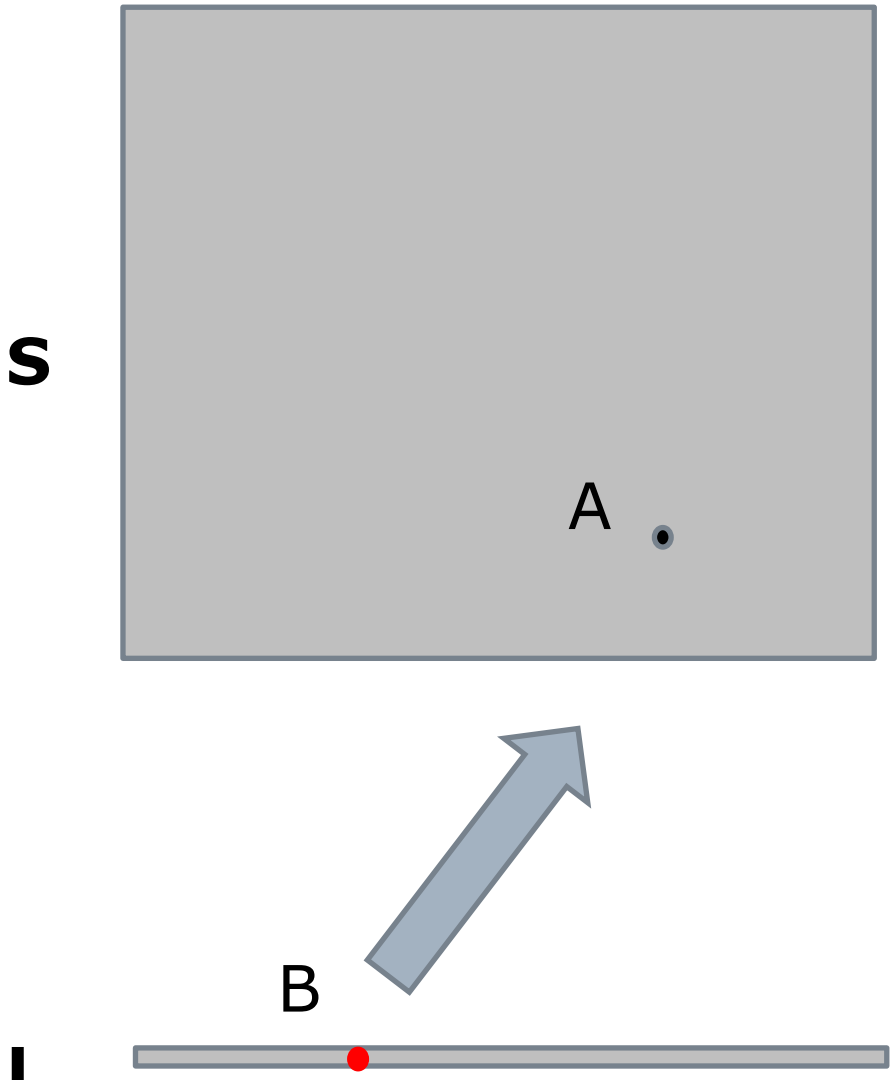
この時、次の実数zを考える

$z=0.x_1y_1x_2y_2x_3y_3x_4y_4x_5y_5\ldots$

このzは、直線上の一点Bに対応する。

こうして、平面上の任意の点を、
直線上の点に対応付けすること
ができる。

平面と直線の点の濃度



直線上のの点Bに対応する実数をzとする。

$z=0.x_1x_2x_3x_4x_5x_6x_7x_8x_9\ldots$

この時、次の実数xとyを考える

$x=0.x_1x_3x_5x_7x_9x_{11}x_{13}x_{15}x_{17}\ldots$

$y=0.x_2x_4x_6x_8x_{10}x_{12}x_{14}x_{16}x_{18}\ldots$

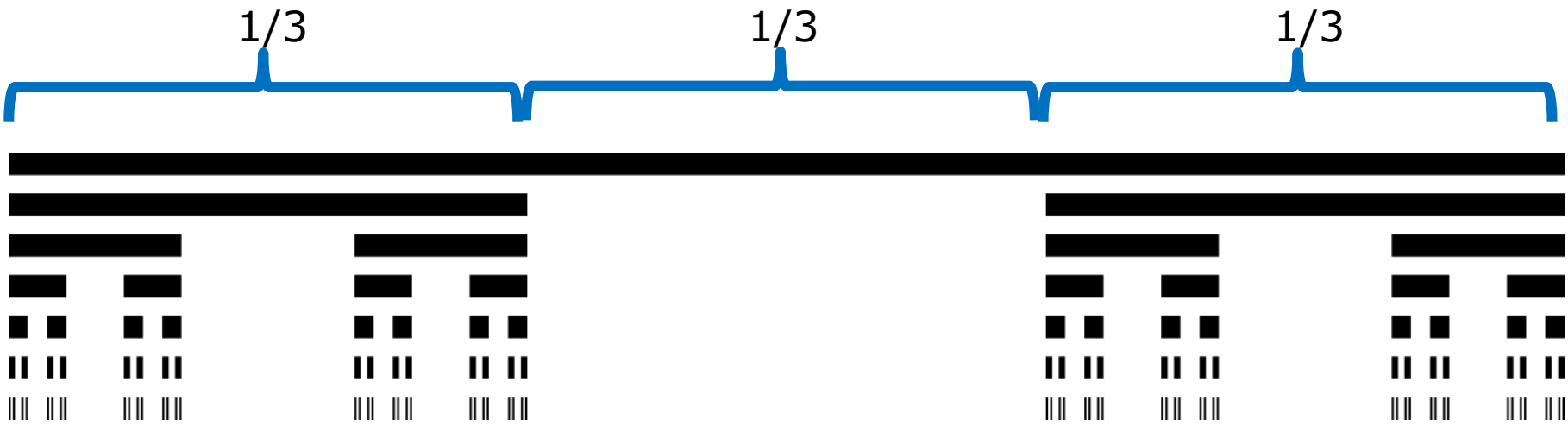
平面上で、(x,y)を座標とする点をAとすれば、直線上の点Bを、平面上の点Aに対応させることができる。

こうして、平面上の点は、直線上の点に一対一で対応する。

カントール集合：奇妙な図形上の点の数

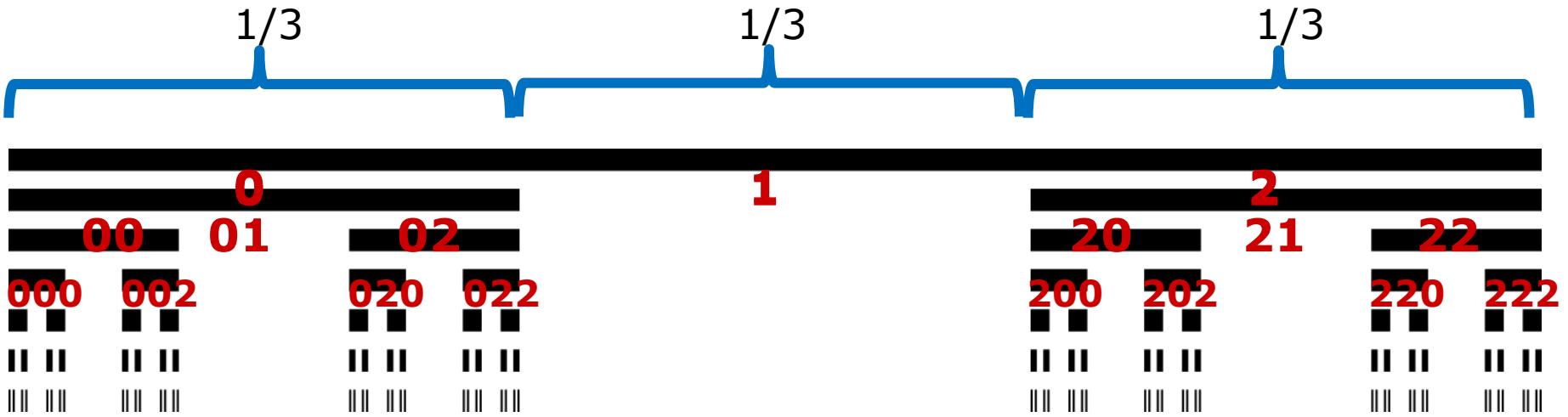
Discovered in 1874 by Henry John Stephen Smith and introduced by Cantor in 1883

カントール集合



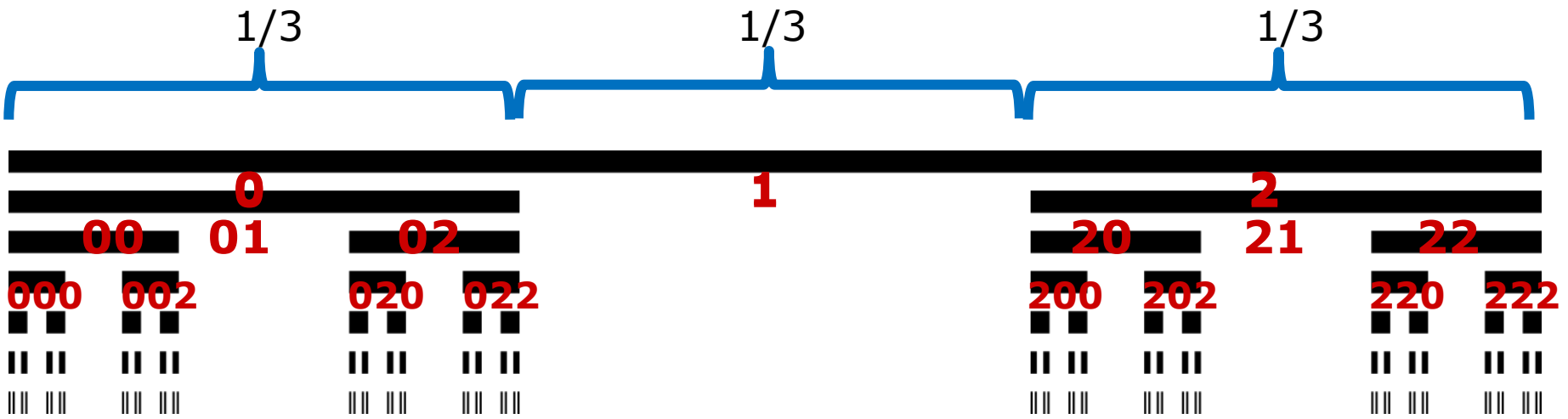
区間[0, 1]を三等分して、真ん中の部分を切り落とす。
左右に残った二つの部分に対しても、真ん中の1/3を
落とす操作を、無限回繰り返して残ったのがカントール
集合。この図では、その操作を6回ほど繰り返している。

カントール集合の表現　0,1,2 の三進数



三等分した区間を、左から **0, 1, 2** とラベルをつける。
一回目の操作で、**1**の部分が落ちて、**0**と**2**の部分が残る。
二回目の操作を残った0と2の部分に同じように行う。そこで残った部分は、**00, 02, 20, 22** というラベルをつけることができる。同様な操作を繰り返す。
結局、カントール集合上の点は、1が現れない0と2からなる無限列で表現されることがわかる。

カントール集合　長さ＝0



このカントール集合は、操作を繰り返すたびに、
長さが1/3,2/3･1/3,4/27,..づつ切り取られていく。
無限回の操作の後には、切り取られた部分の長さは、

$$\frac{1}{3}+\frac{2}{9}+\frac{4}{27}+\frac{8}{81}+\cdots=\frac{1}{3}\left(\frac{1}{1-\frac{2}{3}}\right)=1$$

となる。もとの長さが1だから、この図形の長さは
1-1＝0となる。

カントール集合の濃度

驚くべきことに、この長さ0のカントール集合は、実は、実数と同じ濃度を持っている！　そのことは、次のようにして分かる。
この集合上の点xが、0,1,2の三進表記（ただし、1は現れない。）で、例えば、次のように表現されたとしよう。

x=0.22020022020002022…..

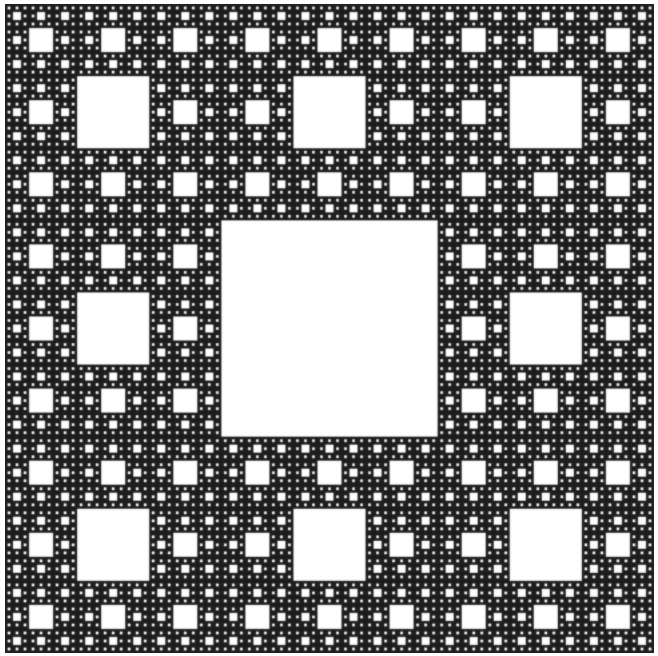
この時、ここで現れる 2 を全て1に置き換えた数をyとする。

y=0.11010011010001011……

xとyは、一対一で、対応している。
このyを、[0,1]区間の実数の二進表記だと考えれば、カントール集合上の点xは、[0,1]区間の実数の一つに対応していることが分かる。逆に、[0,1]区間の実数の二進表記で、1を2に置き換えれば、この実数のカントール集合上の対応する点がわかる。

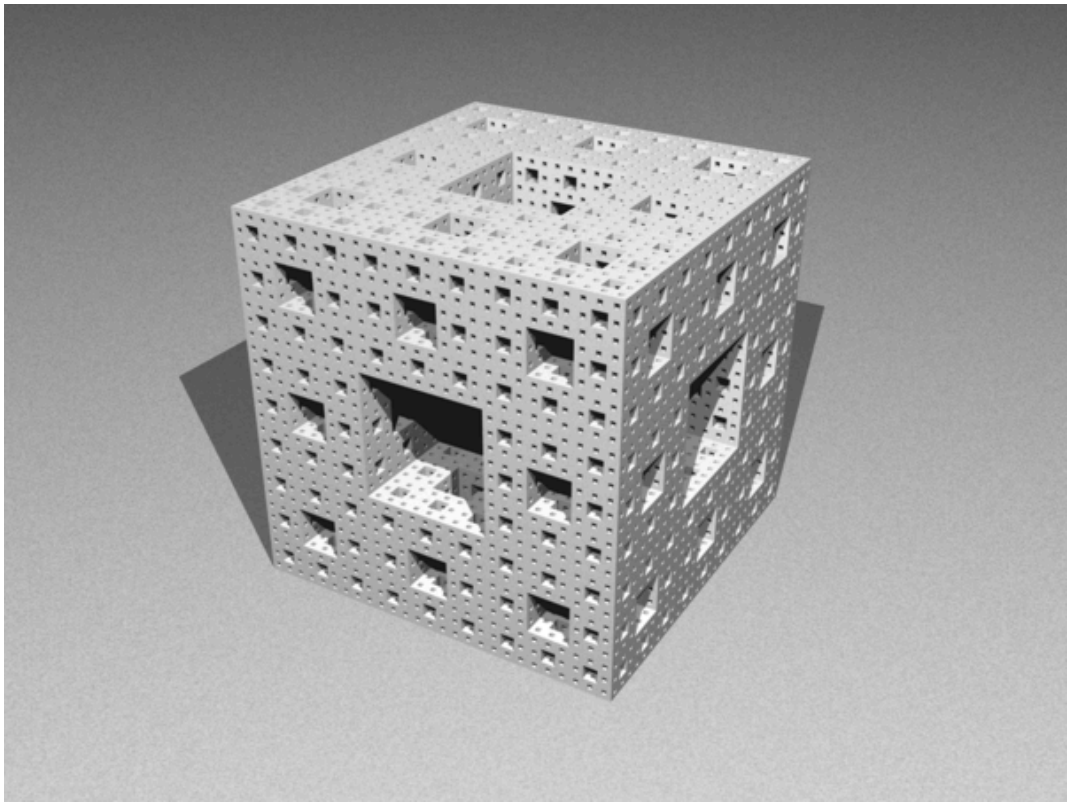
不思議で奇妙な例

カントール集合の二次元版
シェルピンスキーのカーペット



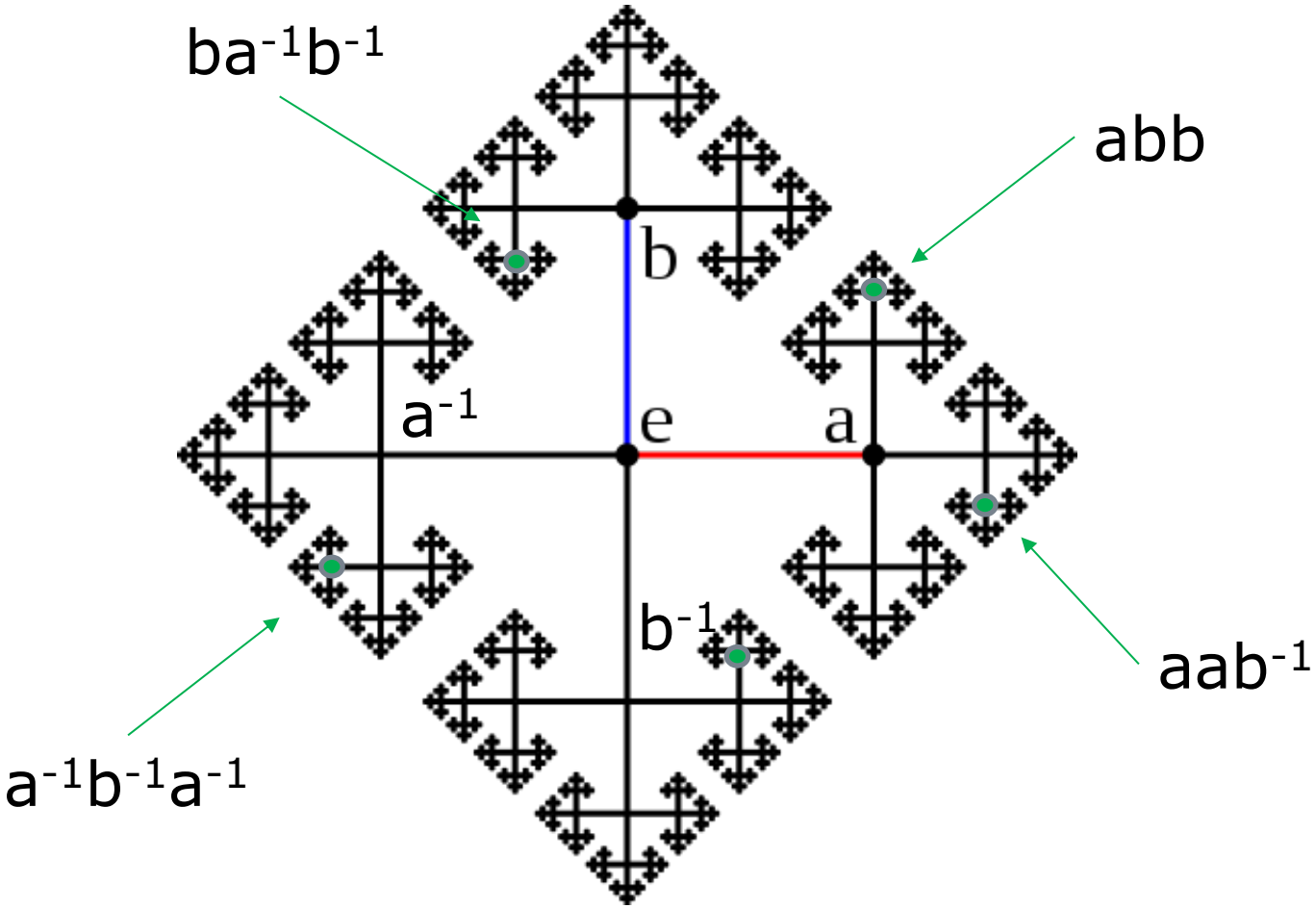
この図形も、測度（面積）ゼロだが、実数と同じ濃度を持つ。

カントール集合の三次元版
メンガーのスポンジ

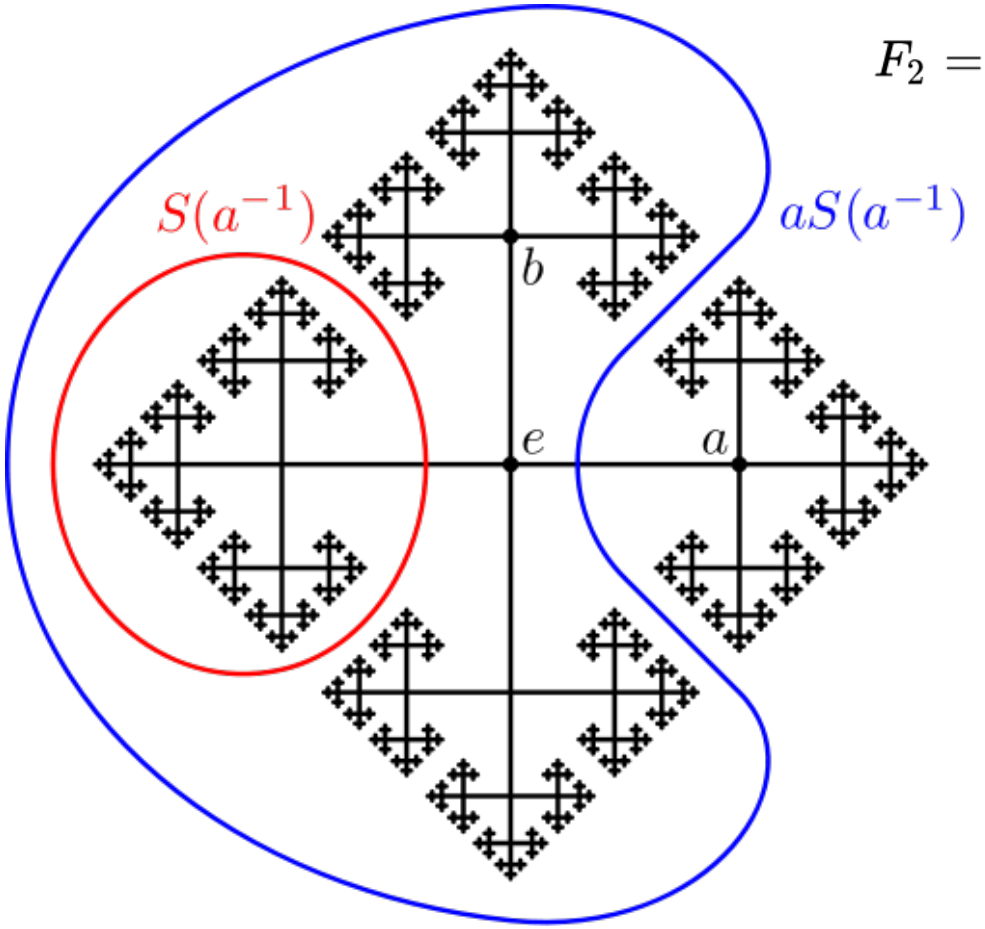


この図形も、測度（体積）ゼロだが、実数と同じ濃度を持つ。
しかし、表面積は無限大である。

F_2のケーリー・グラフ
パラドキシカルな F_2の分割



パラドキシカルな F_2の分割



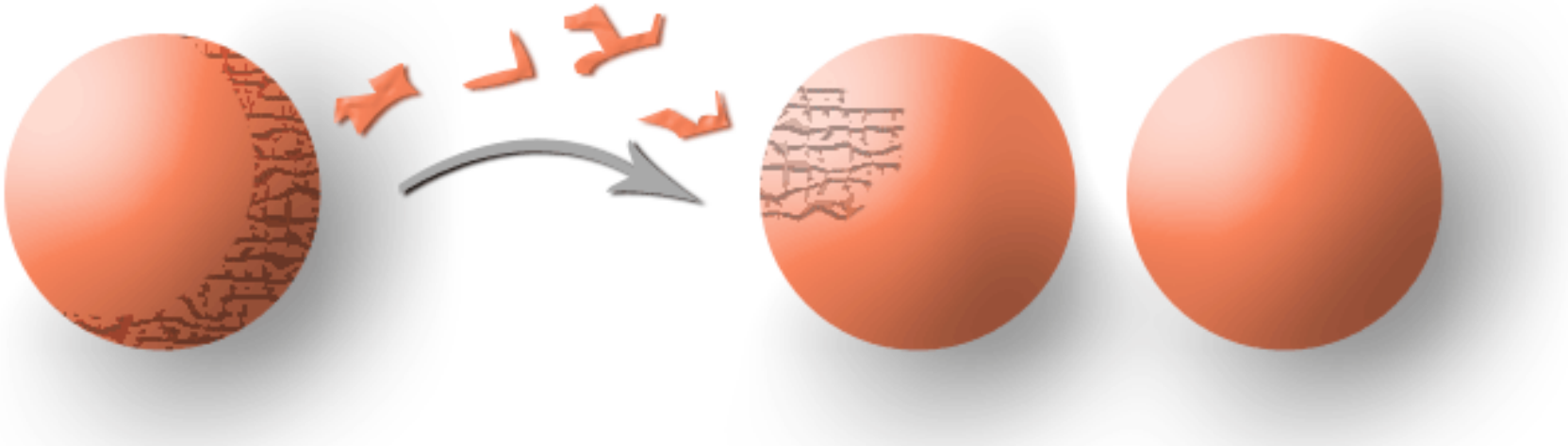
$$F_2 = \{e\} \cup S(a) \cup S(a^{-1}) \cup S(b) \cup S(b^{-1})$$

$$F_2 = aS(a^{-1}) \cup S(a)$$

$$F_2 = bS(b^{-1}) \cup S(b)$$

Banach–Tarski paradox

- 球を3次元空間内で、有限個の部分に分割し、それらを回転・平行移動操作のみを使ってうまく組み替えることで、元の球と同じ半径の球を2つ作ることができるという定理



カントールが証明できなかったこと:
「連続体仮説」

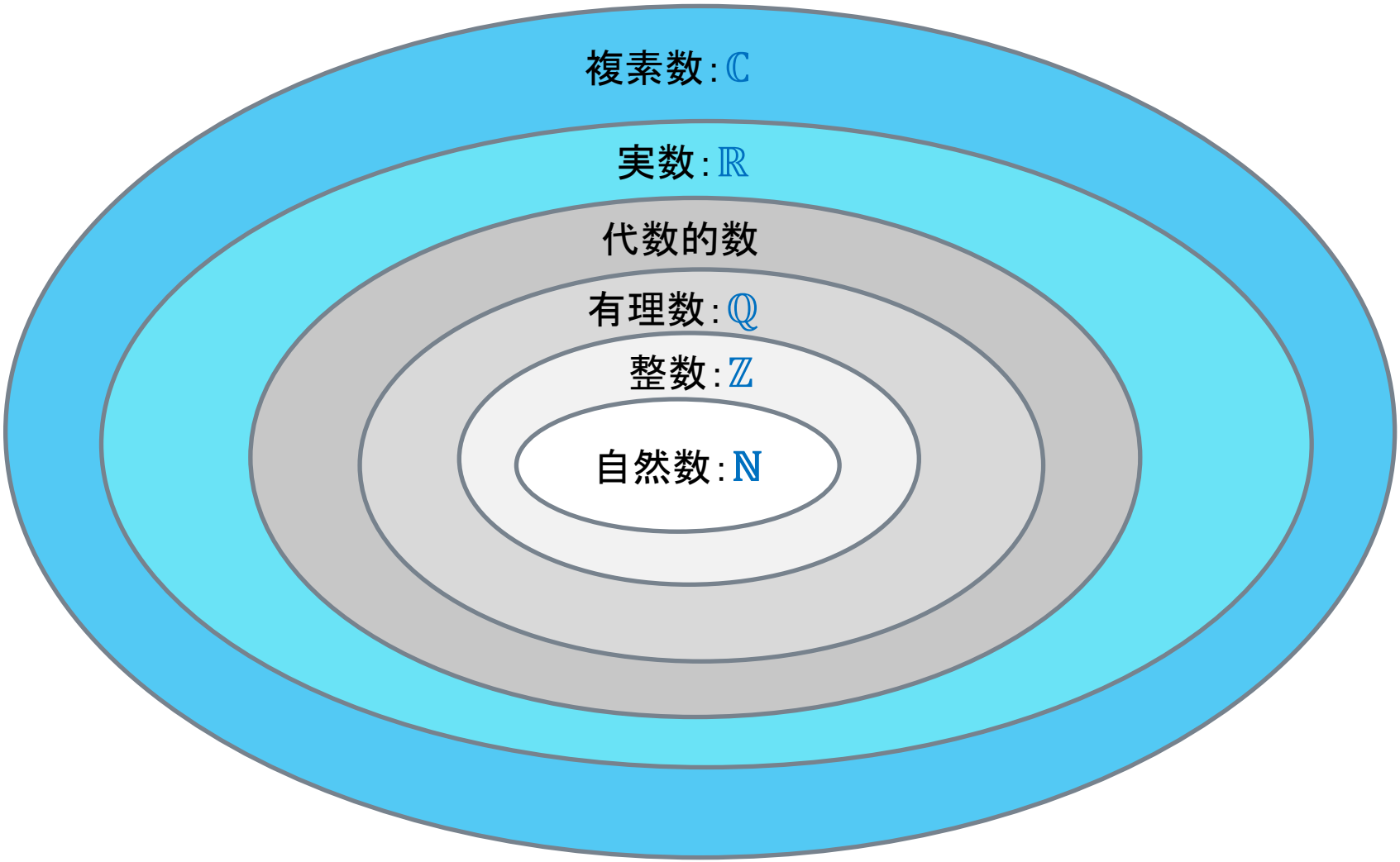
無限の二つの系列

- 可算無限集合の濃度を$\aleph_0$（アレフゼロ）と呼ぶ。
- 集合Aの濃度（要素の数）を|A|で表すとする。
  $|\omega| = \aleph_0$ である。
- 任意の濃度 $\aleph_\alpha$について、$\aleph_\alpha < 2^{\aleph_\alpha}$ を示すことはできるので、無限の濃度にも無限の階層があることがわかる。

- こうして、順序数と基数の二つの無限の系列があることになる。
  有限集合の時、両者は、ある自然数に一致する。

順序数の系列：0,1,2,3,…,ω,…
基数の系列　：0,1,2,3,…,$\aleph_0$,$\aleph_1$,$\aleph_2$,$\aleph_3$,…

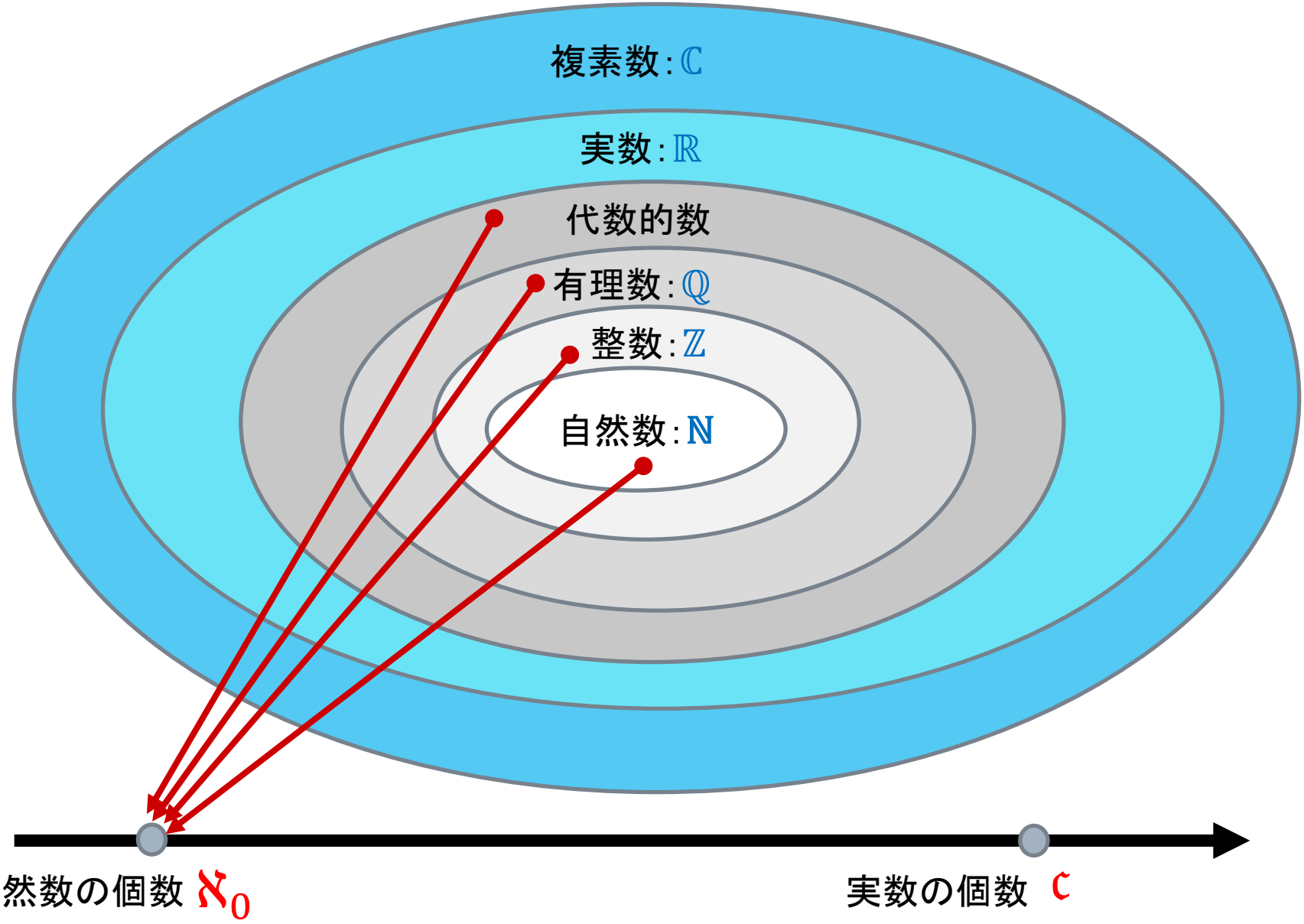
数の階層



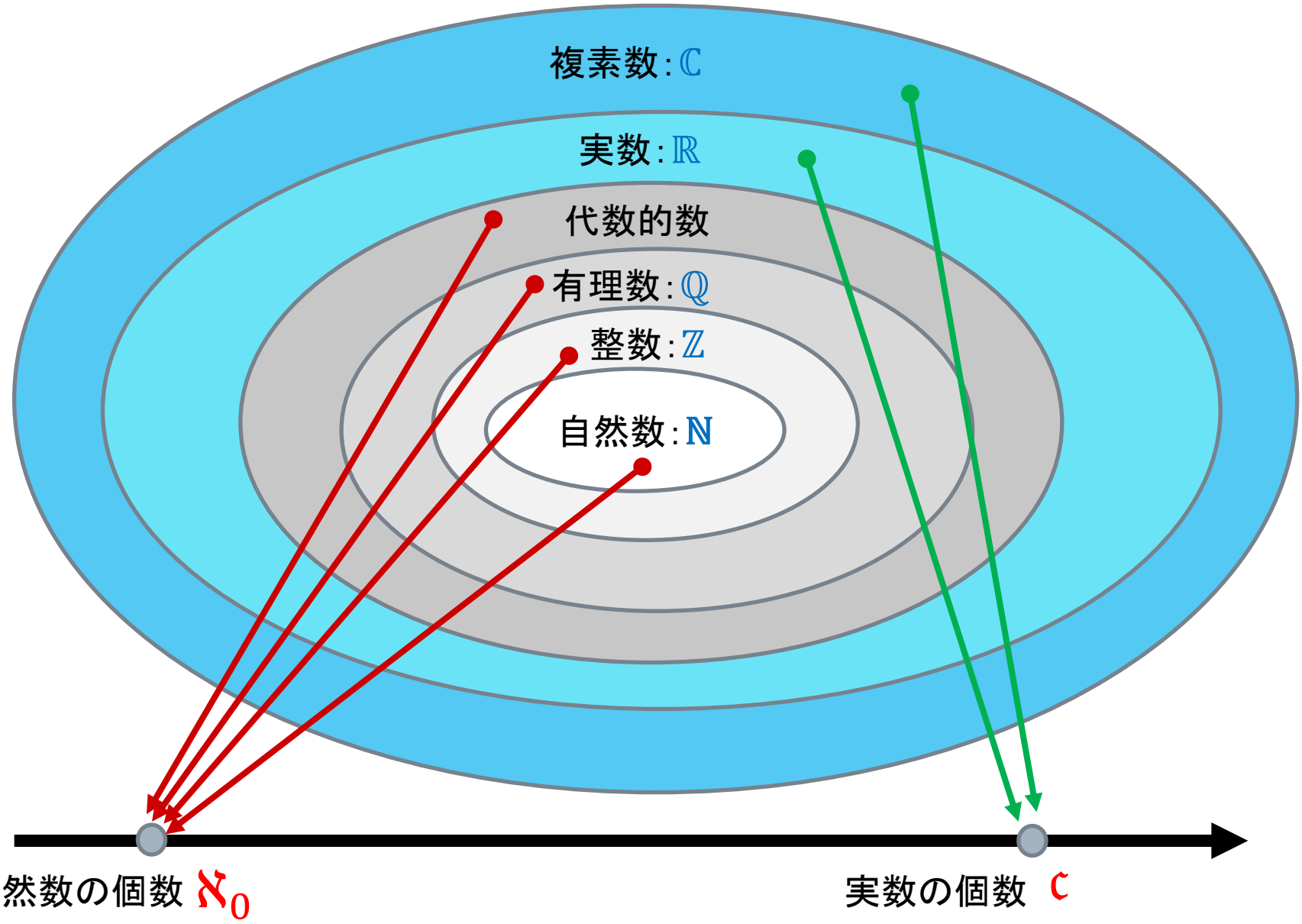
自然数の個数 $\aleph_0$

実数の個数 $\mathfrak{c}$

無限の数の濃度についてカントールが示したこと



無限の数の濃度についてカントールが示したこと



連続体仮説 1878年

- カントールは、彼が見つけた可算無限の濃度$\aleph_0$より大きな実数の無限の濃度 $\mathfrak{c}$を、無限濃度の階層の次の無限$\aleph_1$に等しいと見当をつけた。またこれが、$2^{\aleph_0}$に等しいと予想した。これが、カントールの「連続体仮説 CH(Continuum Hyposesis)」である。

$$\mathfrak{c} = \aleph_1 = 2^{\aleph_0}$$

- さらに、一般に、次の関係が成り立つとした。それを「一般連続体仮説 GCH(General Continuum Hyposesis) 」という

$$\aleph_{n+1} = 2^{\aleph_n}$$

- しかし、彼はこの問題を解くことが出来なかった。

ヒルベルトの23の問題　1900年

- ヒルベルトは、1900年に、20世紀の数学が解くべき23の問題のリストを発表した。その問題提起は、20世紀の数学に少なくない影響を与えた。カントールの「連続体仮説」は、この23の問題の第一番目の問題に取り上げられている。

1. CANTOR'S PROBLEM OF THE CARDINAL NUMBER OF THE CONTINUUM.

- Two systems, i. e , two assemblages of ordinary real numbers or points, are said to be (according to Cantor) equivalent or of equal cardinal number, if they can be brought into a relation to one another such that to every number of the one assemblage corresponds one and only one definite number of the other. The investigations of Cantor on such assemblages of points suggest a very plausible theorem, which nevertheless, in spite of the most strenuous efforts, no one has succeeded in proving.

ヒルベルトの23の問題　1900年

- Every system of infinitely many real numbers, i. e., every assemblage of numbers (or points), is either equivalent to the assemblage of natural integers, 1, 2, 3,… or to the assemblage of all real numbers and therefore to the continuum, that is, to the points of a line ; as regards equivalence there are, therefore, only two assemblages of numbers, the countable assemblage and the continuum.

整列性定理も取り上げられている

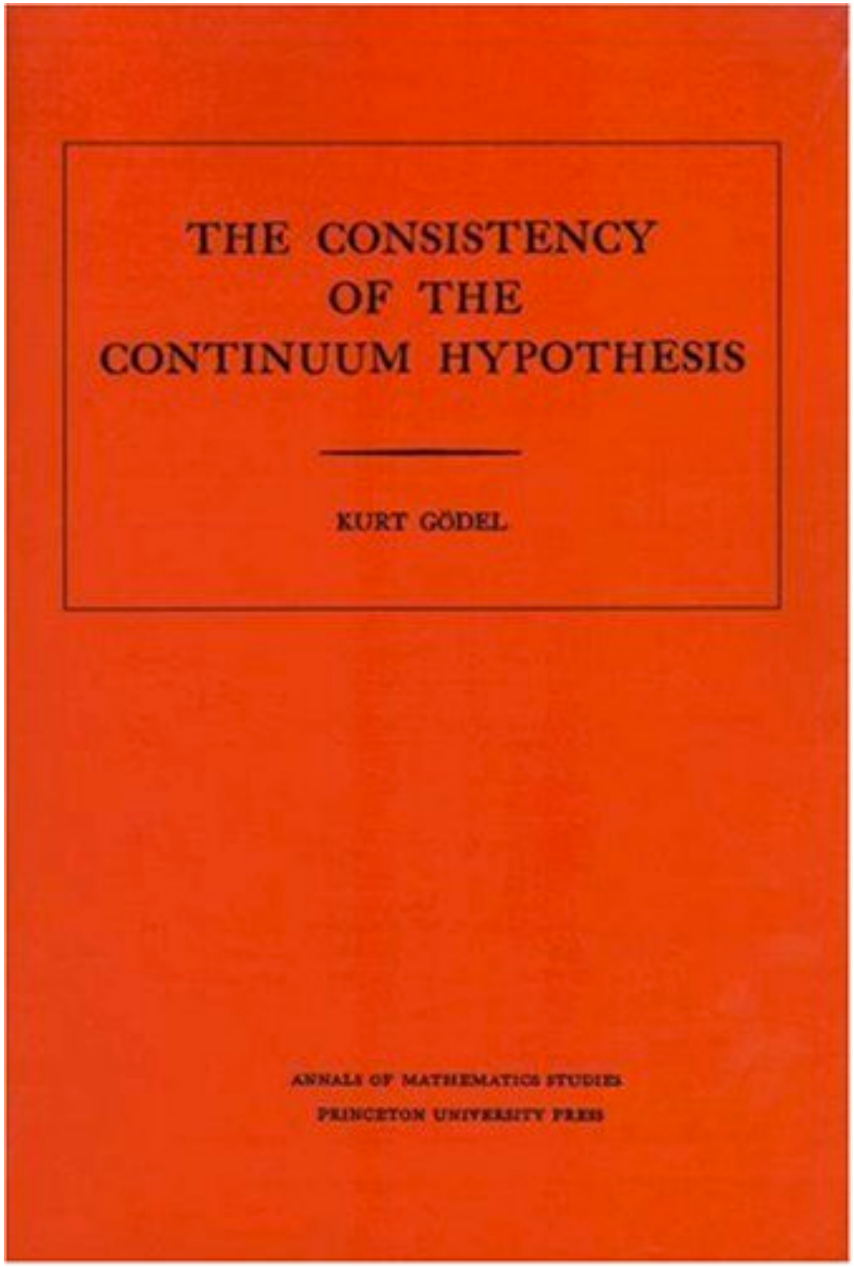
- Let me mention another very remarkable statement of Cantor's which stands in the closest connection with the theorem mentioned and which, perhaps, offers the key to its proof.
- Now Cantor considers a particular kind of ordered assemblage which he designates as a well ordered assemblage and which is characterized in this way, that not only in the assemblage itself but also in every partial assemblage there exists a first number.

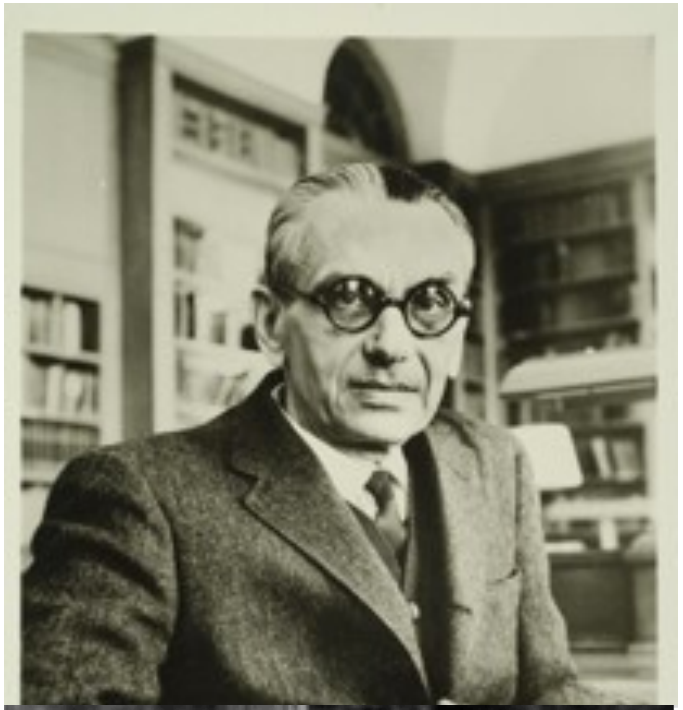
非カントール的集合論の発見

連続体仮説と集合論の無矛盾性の証明
ゲーデル　1940年

ゲーデル
連続体仮説と集合論の無矛盾性の証明

- 「世紀の難問」と呼ばれた連続体仮説の問題で、大きな前進があったのは、1940年のことだ。クルト・ゲーデルが、連続体仮説と選択公理の二つを満たす集合論のモデルを実際に構成して見せたのだ。この集合論のモデルの中では、連続体仮説は「真」である。
- この結果は、連続体仮説を直接に証明したものではないことに注意しよう。それは、集合論と連続体仮説には矛盾がないこと、すなわち、集合論の中では連続体仮説の否定は証明できないことを意味している。


THE CONSISTENCY
OF THE
CONTINUUM HYPOTHESIS
KURT GÖDEL
ANNALS OF MATHEMATICS STUDIES
PRINCETON UNIVERSITY PRESS



ゲーデル 完全性定理 1929年
「無矛盾な理論はモデルを持つ」

- ゲーデルは、ある形式的体系の中に、その証明もその否定の証明も、その形式的体系の中では与えることができない命題Pが存在するという「不完全性定理」（1930年）で有名である。特に、その形式的体系が「無矛盾」であることを主張する命題自身が、この形式的体系では証明できないという発見は、よく知られている。
- ゲーデルには、もう一つ、重要な業績がある。それは「無矛盾な理論はモデルを持つ」というもので、「完全性定理」と呼ばれる。ある形式体系のある命題Pが、その形式的体系のすべてのモデルで妥当する時、その命題Pは証明可能となる。

レーベンハイム・スコーレムの定理
「モデルをもつなら、可算モデルも持つ」

- Skolem's paradox is that every countable axiomatisation of set theory in first-order logic, if it is consistent, has a model that is countable. This appears contradictory because it is possible to prove, from those same axioms, a sentence that intuitively says (or that precisely says in the standard model of the theory) that there exist sets that are not countable. Thus the seeming contradiction is that a model that is itself countable, and which therefore contains only countable sets, satisfies the first order sentence that intuitively states "there are uncountable sets".

連続体仮説と集合論の独立性の証明
コーエン　1963年

コーエン　1963年
連続体仮説と集合論の独立性の証明

- 連続体仮説を最終的に解決したのは、J.P.コーエンだった。彼は、「強制法(Forcing Method)」というモデル構成法を創出し、連続体仮説の否定が成り立つモデルを構成した。
- 先のゲーデルの結果と合わせると、連続体仮説は、集合論の公理群からは独立であることが証明されたことになる。連続体仮説は、集合論の公理群からは証明できないのだ。
- 結果から見ると、カントールが、全力で連続体仮説を証明しようとして、成功しなかったのには理由があったのである。

J.P.Cohen
1934-2007

[19] Gardner, R. S., A. J. Wahba, C. Basilio, R. S. Miller, P. Lengyel, and J. F. Speyer, these PROCEEDINGS, **48**, 2087 (1962).
[20] Ralph, R. K., and H. G. Khorana, *J. Am. Chem. Soc.*, **83**, 2926 (1961).
[21] Ames, B. N., and D. T. Dubin, *J. Biol. Chem.*, **235**, 769 (1960).
[22] Chen, P. S., Jr., T. Y. Toribara, and H. Warner, *Anal. Chem.*, **28**, 1756 (1956).
[23] Heppel, L. A., D. R. Harkness, and R. J. Hilmoe, *J. Biol. Chem.*, **237**, 841 (1962).
[24] Russell, W. E., and H. G. Khorana, *J. Biol. Chem.*, **234**, 2114 (1959).
[25] Kay, E. R. M., N. S. Simmons, and A. L. Dounce, *J. Am. Chem. Soc.*, **74**, 1724 (1952).
[26] Sherman, J. R., and J. Adler, *J. Biol. Chem.*, **238**, 873 (1962).
[27] Waley, S. G., and J. Watson, *Biochem. J.*, **55**, 328 (1953).
[28] Akabori, S., K. Ohno, and K. Narita, *Bull. Chem. Soc. Japan*, **25**, 214 (1952).
[29] Lowry, O. H., N. J. Rosebrough, A. L. Farr, and R. J. Randall, *J. Biol. Chem.*, **193**, 265 (1951).
[30] Nakamoto, T., and S. B. Weiss, these PROCEEDINGS, **48**, 880 (1962).
[31] Fox, C. F., W. S. Robinson, and S. B. Weiss, *Fed. Proc.*, **22**, 463 (1963).
[32] Krakow, J. S., and S. Ochoa, these PROCEEDINGS, **49**, 88 (1963).

THE INDEPENDENCE OF THE CONTINUUM HYPOTHESIS

BY PAUL J. COHEN*

DEPARTMENT OF MATHEMATICS, STANFORD UNIVERSITY

Communicated by Kurt Gödel, September 30, 1963

This is the first of two notes in which we outline a proof of the fact that the Continuum Hypothesis cannot be derived from the other axioms of set theory, including the Axiom of Choice. Since Gödel[1] has shown that the Continuum Hypothesis is consistent with these axioms, the independence of the hypothesis is thus established. We shall work with the usual axioms for Zermelo-Fraenkel set theory,[2] and by Z-F we shall denote these axioms without the Axiom of Choice, (but with the Axiom of Regularity). By a model for Z-F we shall always mean a collection of actual sets with the usual ϵ-relation satisfying Z-F. We use the standard definitions[3] for the set of integers ω, ordinal, and cardinal numbers.

THEOREM 1. *There are models for Z-F in which the following occur:*

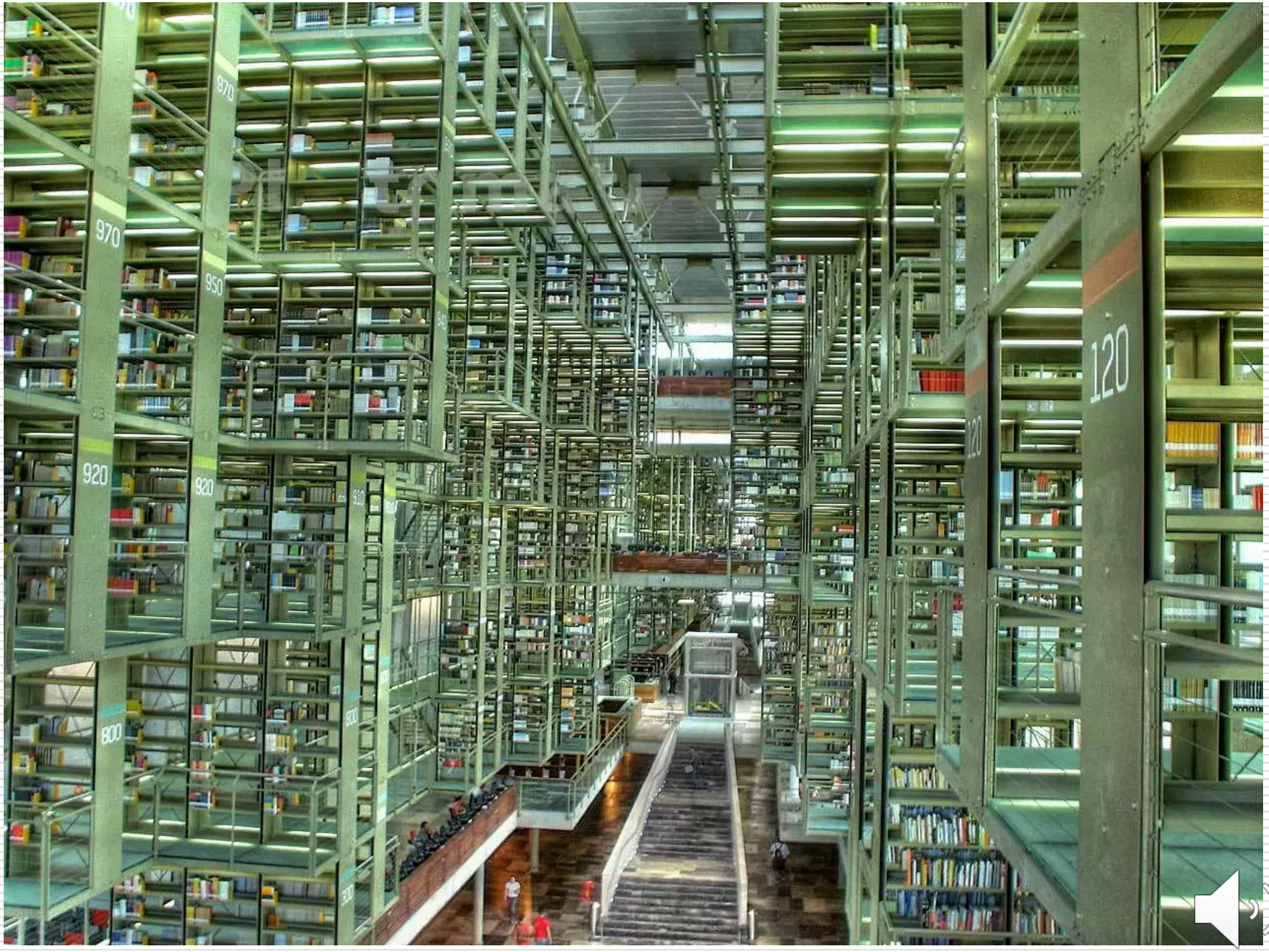
(1) *There is a set a, $a \subseteq \omega$ such that a is not constructible in the sense of reference 3, yet the Axiom of Choice and the Generalized Continuum Hypothesis both hold.*
(2) *The continuum (i.e., $\mathcal{P}(\omega)$ where $\mathcal{P}$ means power set) has no well-ordering.*
(3) *The Axiom of Choice holds, but $\aleph_1 \neq 2^{\aleph_0}$.*
(4) *The Axiom of Choice for countable pairs of elements in $\mathcal{P}(\mathcal{P}(\omega))$ fails.*

Only part 3 will be discussed in this paper. In parts 1 and 3 the universe is well-ordered by a single definable relation. Note that 4 implies that there is no simple ordering of $\mathcal{P}(\mathcal{P}(\omega))$. Since the Axiom of Constructibility implies the Generalized Continuum Hypothesis,[3] and the latter implies the Axiom of Choice,[3] Theorem 1 completely settles the question of the relative strength of these axioms.

Before giving details, we sketch the intuitive ideas involved. The starting point is the realization[1, 4] that no formula $\alpha(x)$ can be shown from the axioms of Z-F to have the property that the collection of all x satisfying it form a model for Z-F in which the Axiom of Constructibility ($V = L$,[3]) fails. Thus, to find such models, it seems natural to strengthen Z-F by postulating the existence of a set which is a

非カントール的集合論の発見

- コーエンの結果は、始祖のカントールが構想した、いわばカントール的集合論とは異なる、非カントール的集合論が存在することを明らかにした衝撃的なものだった。
- ユークリッド幾何学の「第五公準」が、他のユークリッド幾何学の公理群からは独立で、それらから証明できないことの認識の成立が、非ユークリッド幾何学を成立させたのと同じことが、「数学の基礎」である集合論で起きたのである。
- コーエンの結果とその方法は、1960年代の中頃には、集合論の構造についての認識を飛躍的に発展させた。


970
950
920
800
120

Turing Machine 入門

Agenda
Turing Machine入門

- チューリング・マシンとは何か？
- チューリング・マシンのサンプル
  - 例1： 文字列の長さを求める
  - 例2： 文字列中の'1'の数の偶奇を求める
  - 例3： 文字列のコピー
  - 例4： 括弧のバランスのチェック
- チューリング・マシンとコンピュータのプログラム
  - 例5： 無限ループ
  - 例6： Goto L1, Goto L2
  - 例7： 条件分岐

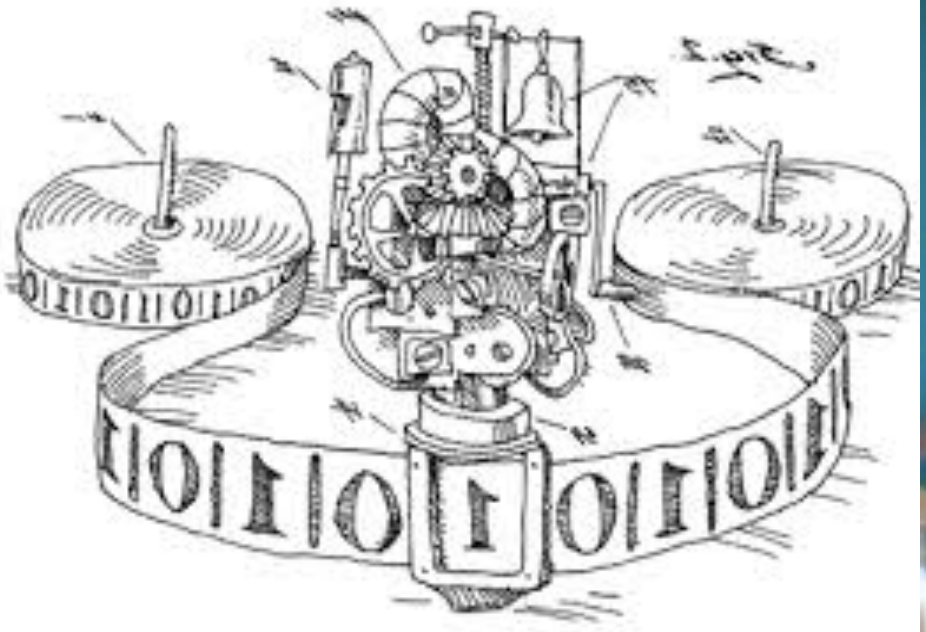
チューリング・マシンとは何か？

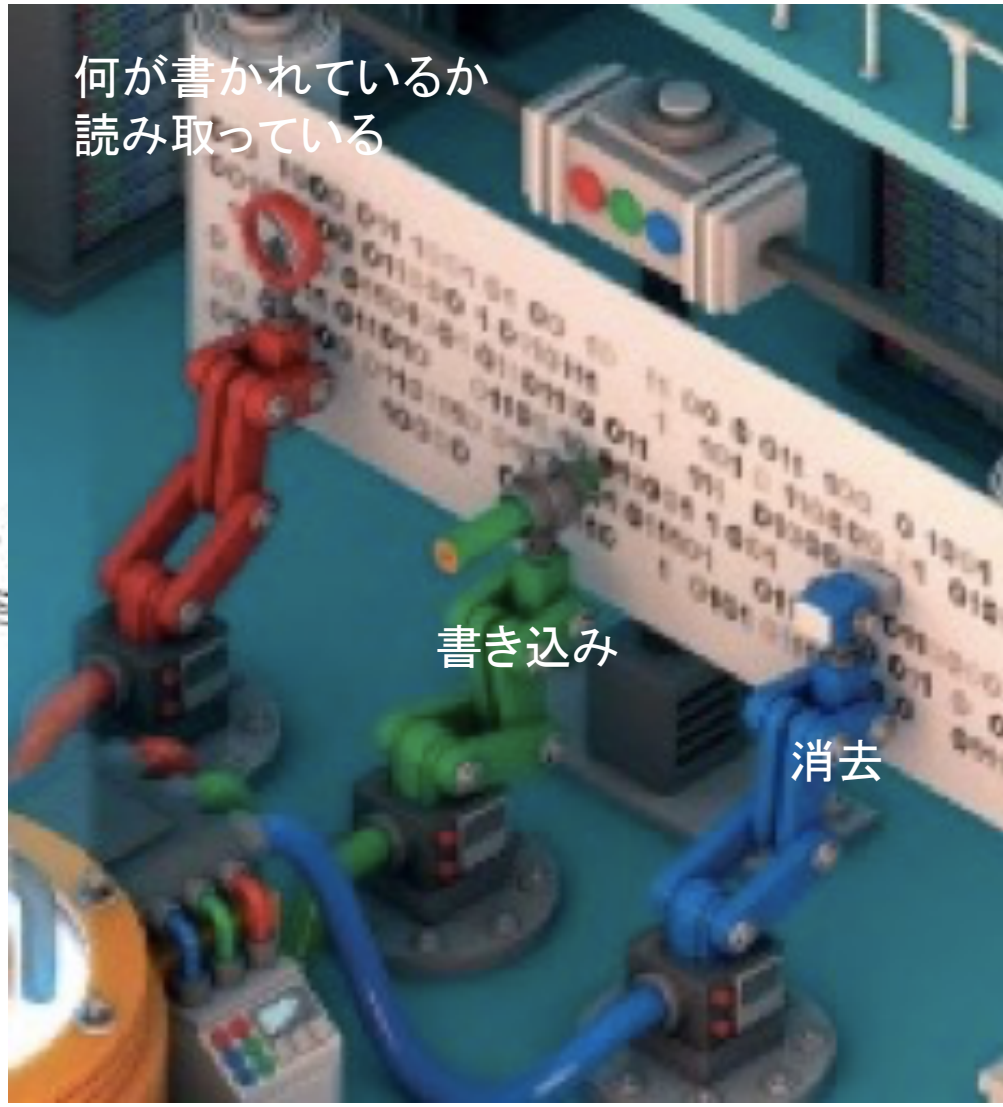
“ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM”

Alan Turing　1936年

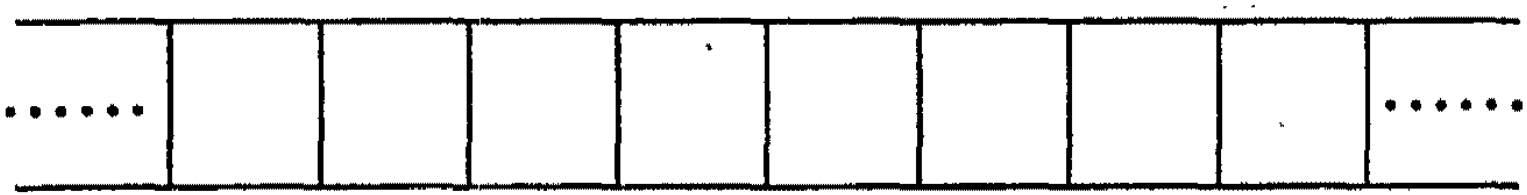
https://goo.gl/mi4MBb

チューリング・マシンのイメージ

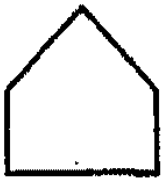




チューリング・マシンのメカニズム



テープ（マスで区切られている）



ヘッド

テープ：ひとマスずつに区切られており、左右に移動する。長い。

ヘッド：テープのひとマスの記号を読み取り、マシンの「状態」に応じて、次の動作をする。

可能な動作：

1. マス目に記号を書き込む（消す＋書く）。あるいは、そのままにして何も書き込まない。
2. マシンの状態を変える。あるいは状態を同じに保つ。
3. ヘッドを、右あるいは左に移動する。（テープが動く）

チューリング・マシンのプログラムの例

Turingマシンのプログラムの例を示す。この表の、例えば、最初の State Aの行で、On '0' の欄の 'B1R' は、「状態Aで、ヘッドが'0'の上にあるなら、状態をBに変えて、1を書き込んで、ヘッドを右（R）に移動する」という命令を表す。

State	on 0	on 1	on 0			on 1		
			Print	Move	Goto	Print	Move	Goto
A	B1R	D0L	1	right	B	0	left	D
B	C1R	F0R	1	right	C	0	right	F
C	C1L	A1L	1	left	C	1	left	A
D	E0L	H1L	0	left	E	1	left	H
E	A1L	B0R	1	left	A	0	right	B

プログラム本体

プログラムの説明

Hは特別な「状態」で、この状態の時、マシンは「停止」する

チューリング・マシンは、
計算可能なすべての計算のモデル

「チャーチ＝チューリングの提言」によれば、計算可能なすべての計算は、チューリング・マシンで計算できることになる。その割には、チューリング・マシンのメカニズムは、驚くほど単純なように見える。

それは、チューリング・マシンが計算の本質を、見事に捉えているからなのだが、このマシンの振る舞いは、見かけ以上に複雑である。

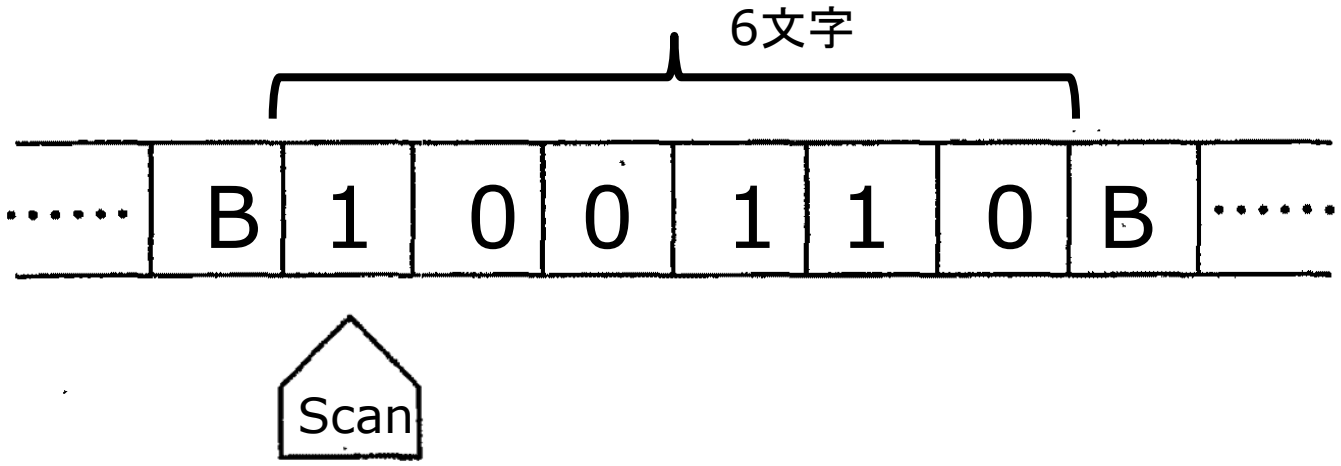
A,B,C,D,E,Fという6つの「状態」を持ち、'0','1'に対応した場合の12個のプログラムで定義されただけの次の単純なチューリング・マシンが、極めて複雑な振る舞いをすることを、あとで紹介しようと思う。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC

チューリング・マシンのサンプル

例1：　文字列の長さを求める

問題1：B(Blank:何も書かれていない空白のマス目）とBのあいだにある、0と1の記号の列（文字列）の長さを求めよ。



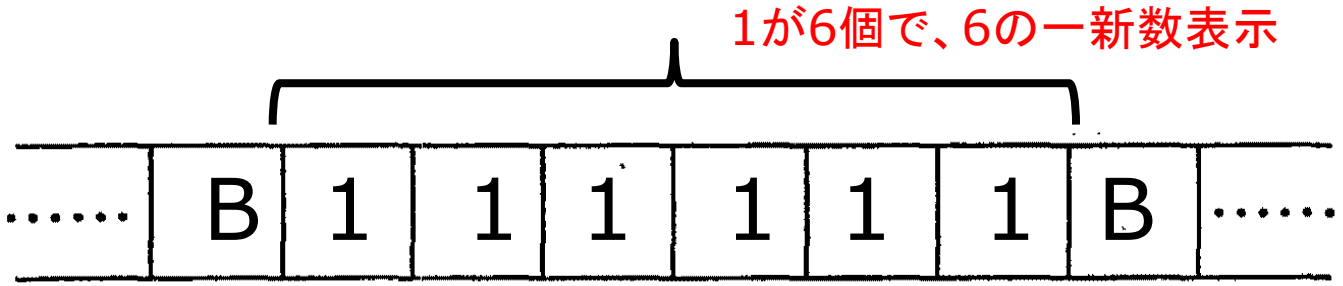
この例なら、6を返せばいい。

例1： 文字列の長さを求める

これは、状態一つでも解ける。この状態をScanとしよう。

状態Scanは、

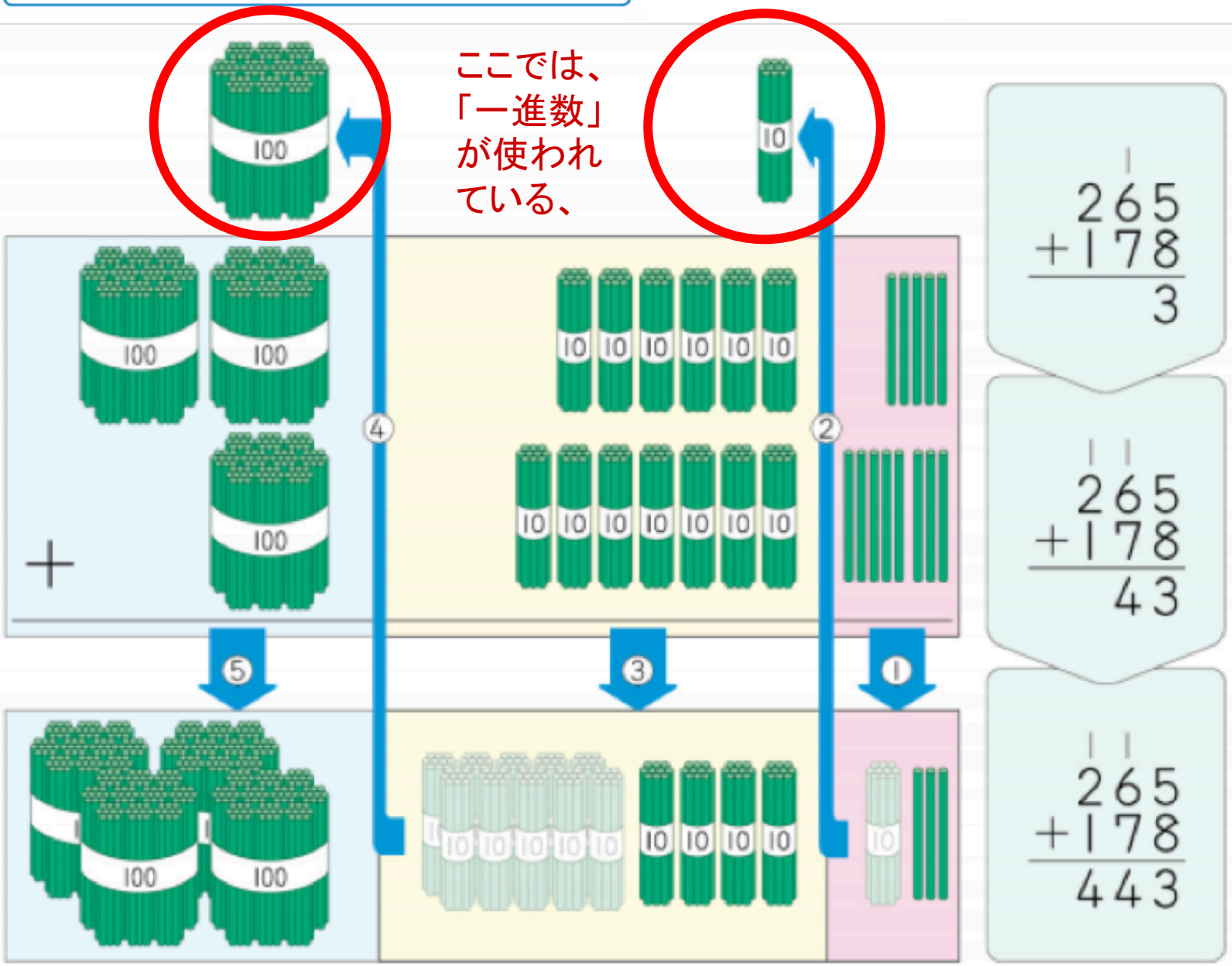
1. `1'の上だと何もしないで、ヘッドを右に進める。状態はそのまま。
2. `0'の上だと`0'を`1'に書き換えて、ヘッドを右に進める。状態はそのまま。
3. `B'の上だと、状態を'HALT'に変えて、停止する。



こんな状態で停止する。実は、これでいいのだ。
これは、答え'6'の「一進数」表示だ！

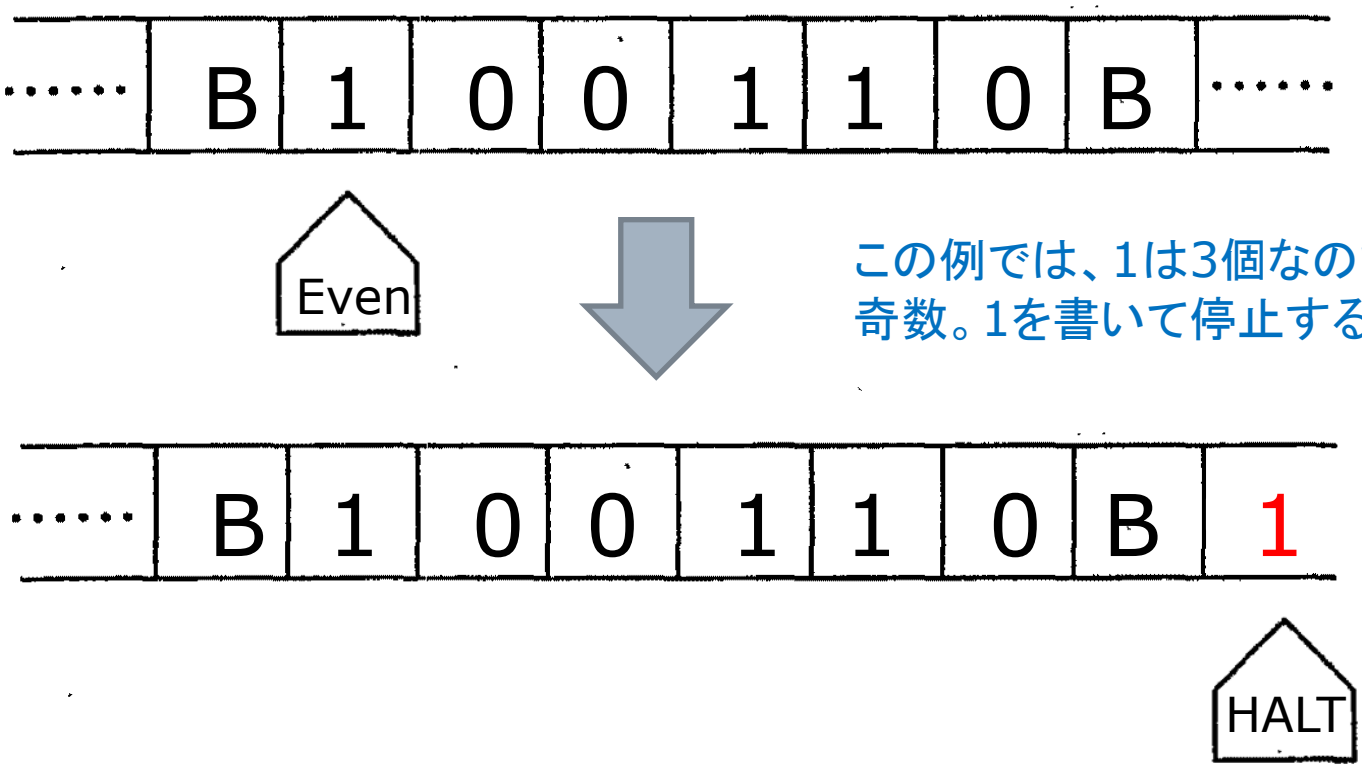
3 265＋178 の筆算のしかたを考えましょう。

啓林館 3年上 p.57



例2:　文字列中の'1'の数の偶奇を求める

問題2:BとBのあいだにある、0と1の文字列中の1の数の偶奇を求めよ。偶数なら0を奇数なら1を返せ。



例2： 文字列中の'1'の数の偶奇を求める

二つの状態 Even と Odd を使う。初期状態をEvenとする。

状態Evenの時、

1. `0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. `1'の上だと何も印字しないで、ヘッドを右に進める。状態はOddに。

状態Oddの時、

1. `0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. `1'の上だと何も印字しないで、ヘッドを右に進める。状態はEvenに。

ただ、これだと文字列末の'B'にヘッドは来た時の動作が定義されていないし、結果の印字もできていない。

新たに、二つの状態、PrintEvenとPrintOddを追加する。

例2： 文字列中の'1'の数の偶奇を求める

状態Evenの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はOddに。
3. 'B'の上だと何も印字しないで、ヘッドを右に進める。状態はPrintEvenに。

状態Oddの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はEvenに。
3. 'B'の上だと何も印字しないで、ヘッドを右に進める。状態はPrintOddに。

状態PrintEvenの時、

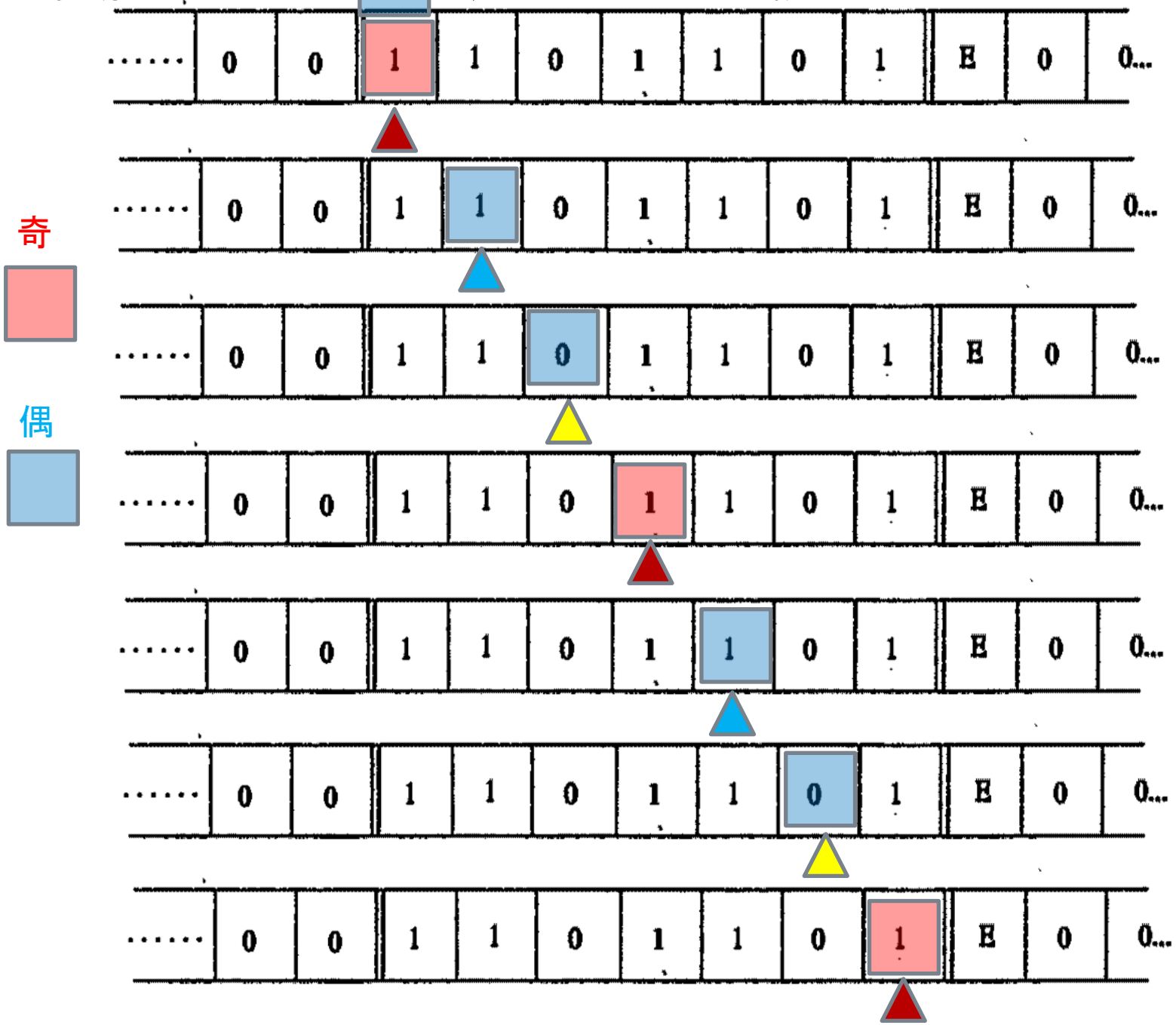
1. 0を印字して停止する。

状態PrintOddの時、

1. 1を印字して停止する。

初期状態＝'偶'

表示されているのは読み取り後のマシンの状態



例3: 文字列のコピー

問題3:AとBとの間にある、0と1からなる文字列をBCの間にコピーする。

A011011BC --> A011011B011011C

A101010BC --> A101010B101010C

A1011BC --> A1011B1011C

A101BC --> A101B101C

例3: 文字列のコピー

- 状態:Fetch /*文字列から一文字読み取る。*/
  - ‘0’であれば、それを’X’に書き換え、状態をCopy0に変え、右に進む
  - ‘1’であれば、それを’Y’に書き換え、状態をCopy1に変え、右に進む
  - ‘B’であれば、コピー終了　/ それ以外なら、状態を変えず、右に進む
- 状態: Copy* /* ヘッドを’C’が見つかるまで右移動する。*/
  - ‘C’以外なら、状態を変えず、右に進む
  - ‘C’で、状態がCopy0であれば、その位置に’0’を書き込み、状態をMark0に変え、右に進む
  - ‘C’で、状態がCopy1であれば、その位置に’1’を書き込み、状態をMark1に変え、右に進む
- 状態:Mark* /* ’C’を書き込み、状態をBack0 / Back1 に変える。 */(略)
- 状態:Back* /* ヘッドを、’X’あるいは’Y’が見つかるまで戻す。*/
  - ‘X’, ‘Y’ 以外なら、左に一つ進む
  - ‘X’で、状態がBack0であれば、’X’の位置に’0’を書き込み、状態をFetchに変えて、右に一つ進む
  - ‘Y’で、状態がBack1であれば、’Y’の位置に’1’を書き込み、状態をFetchに変えて、右に一つ進む

A101BC --> A101B101C

```
Fetch:    A101BC
Copy1:    AX01B1
Mark1:    AX01B1C
Back1:    A101B1C
Fetch:    A101BC
Copy0:    A1Y1B10
Mark0:    A1Y1B10C
Back0:    A101B10C
Fetch:    A101B10C
Copy1:    A10XB101
Mark1:    A10XB101C
Back1:    A101B101C
Fetch:    A101B101C
Halt
```

例4:　括弧のバランスのチェック

問題4: EとEとの間にある、左括弧 `(` と右括弧　`)`のバランスをチェックせよ。

正しくないバランスの括弧の例

E(()()(()))(()E

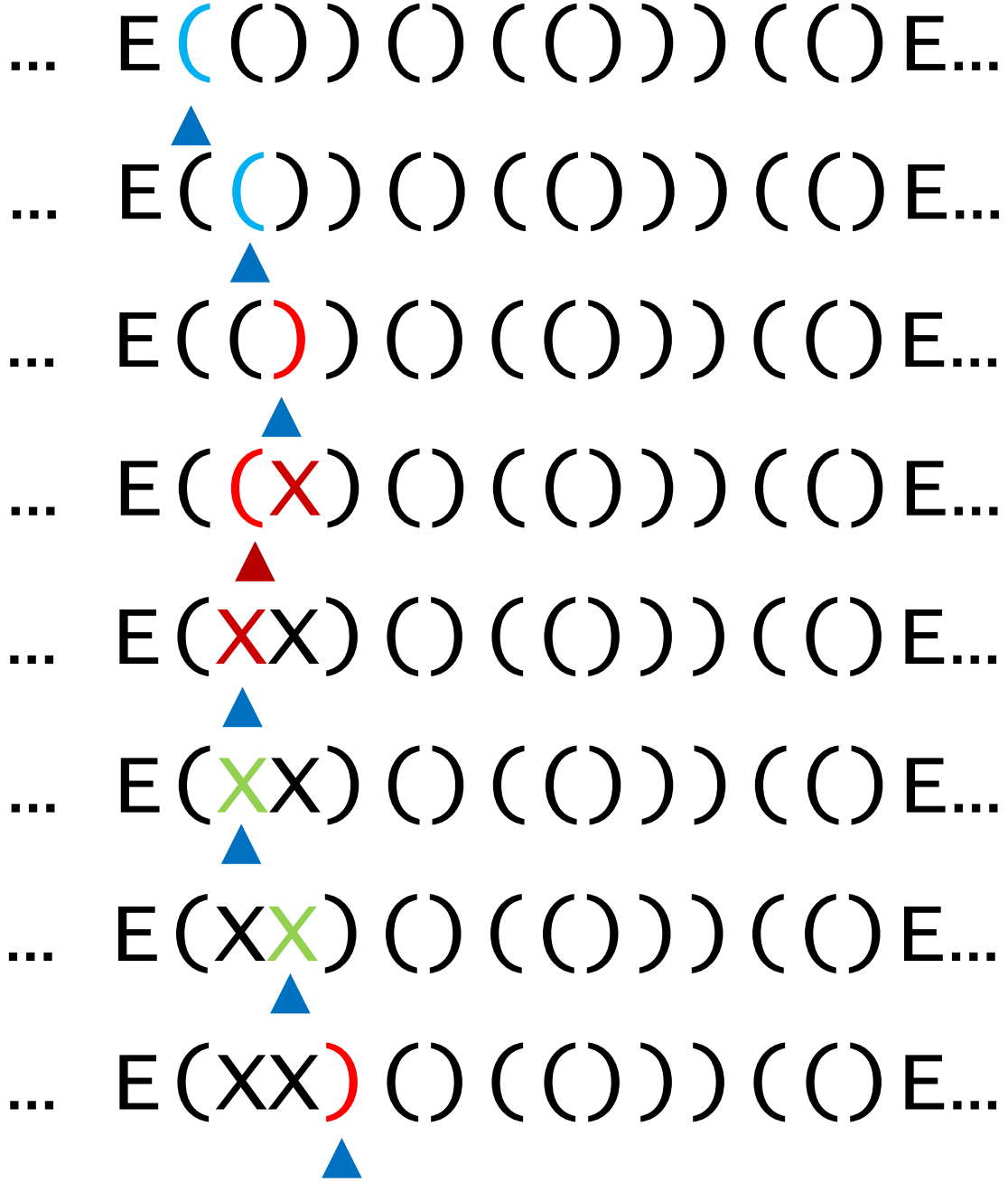
正しいバランスの括弧の例

E(()()(())(())E

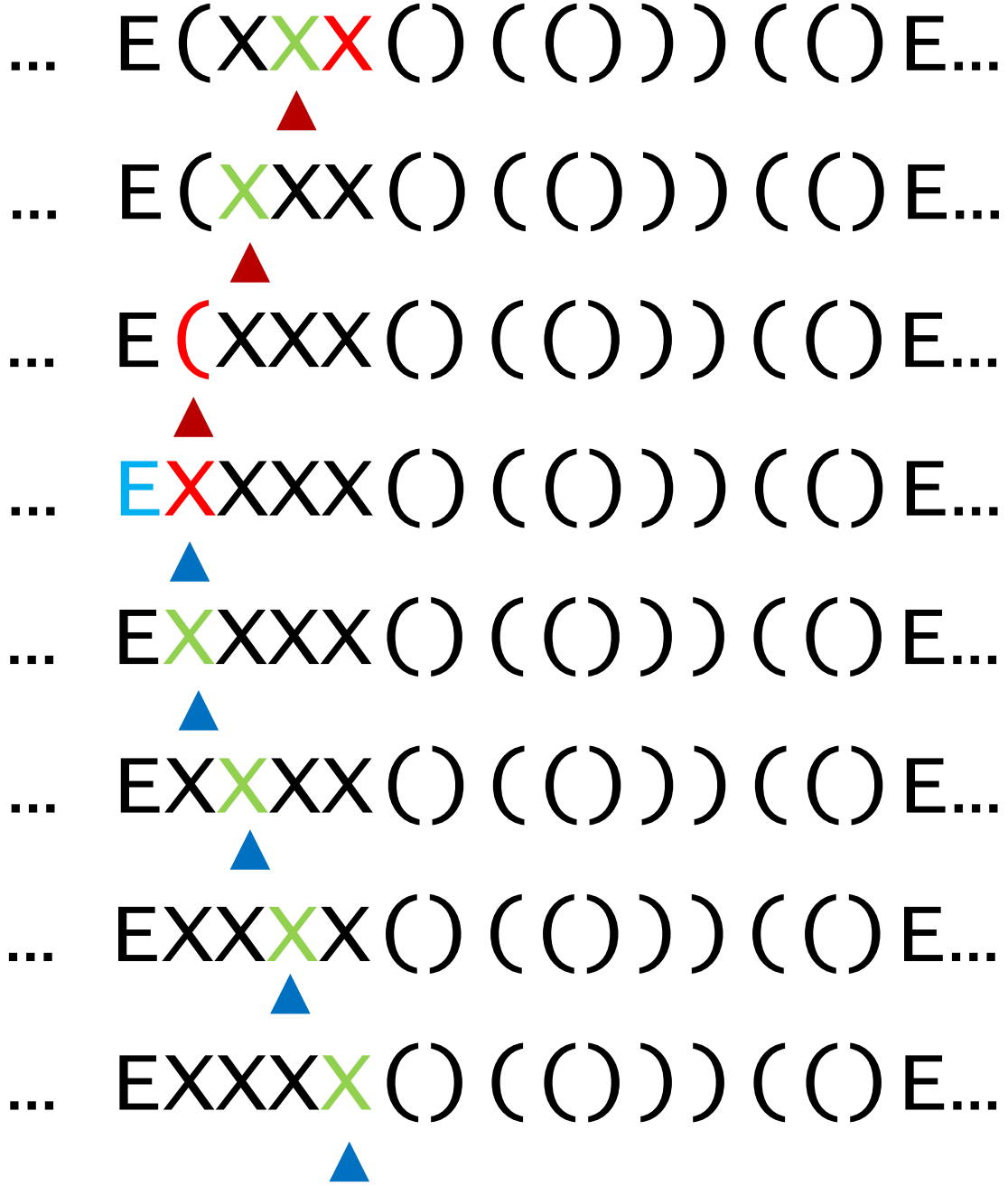
基本的な考え方

1. 右に進んで最初に会った右括弧をXに書き換える。
2. 今度は左に進んで最初に会った左括弧をXに置き換える。
3. また、1.の手順を繰り返す。
4. 全てが、Xになっていれば、正しいバランス。そうでなければ正しくないバランス。

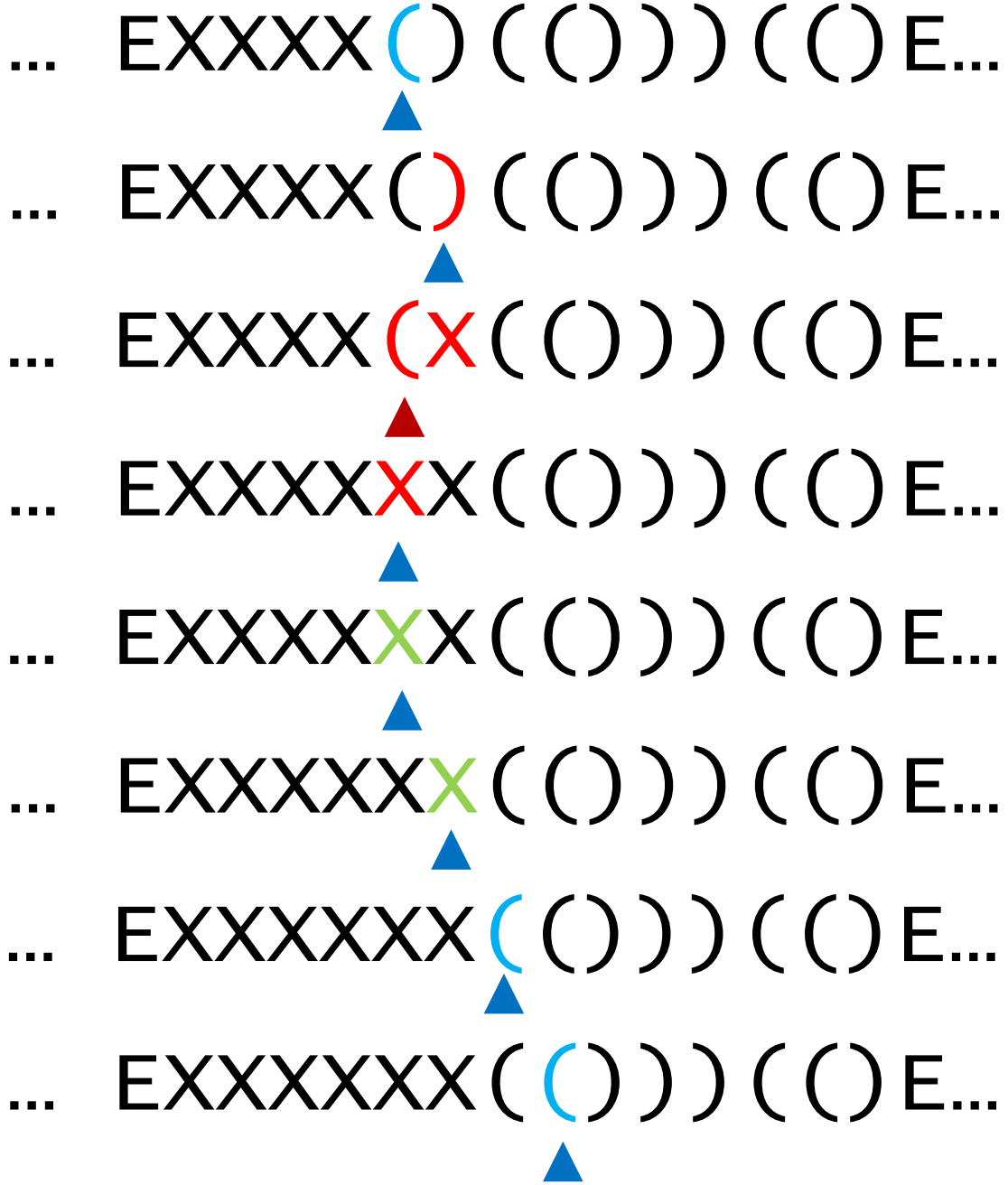
正しくない場合



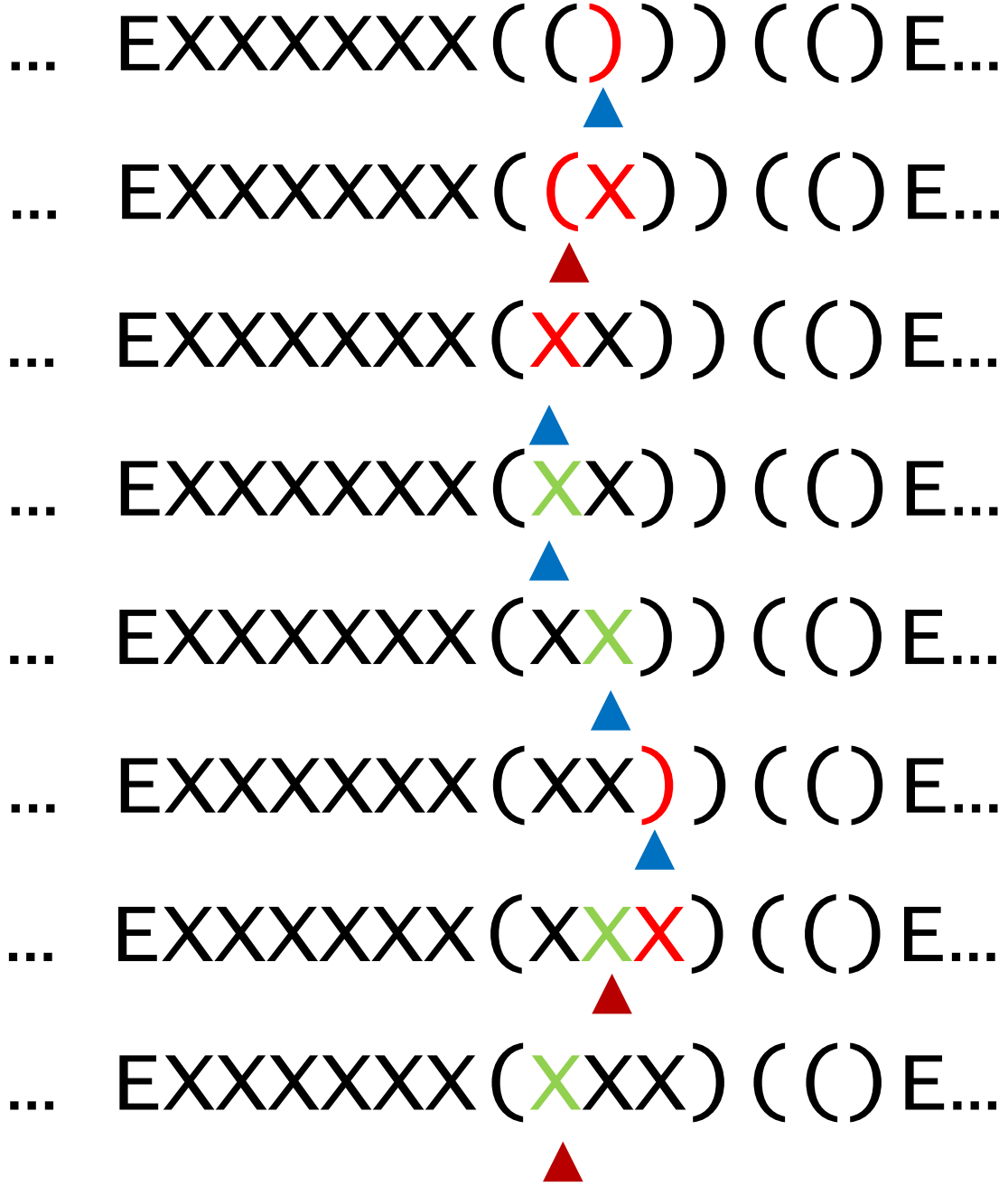
正しくない場合



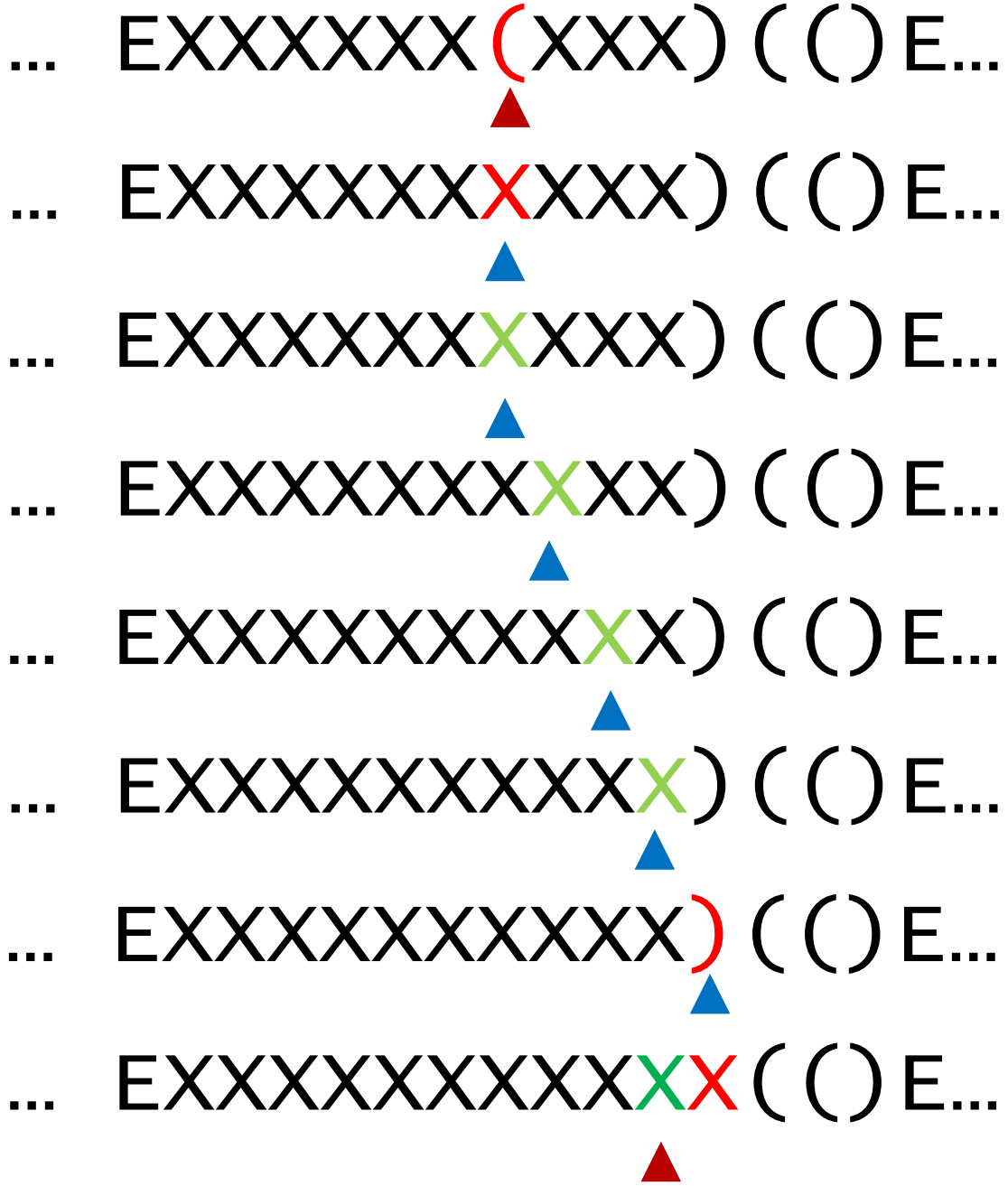
正しくない場合


... EXXXX()(()))(()E...
... EXXXX()(()))(()E...
... EXXXX(X(()))(()E...
... EXXXXXX(()))(()E...
... EXXXXXX(()))(()E...
... EXXXXXX(()))(()E...
... EXXXXXX(()))(()E...
... EXXXXXX(()))(()E...

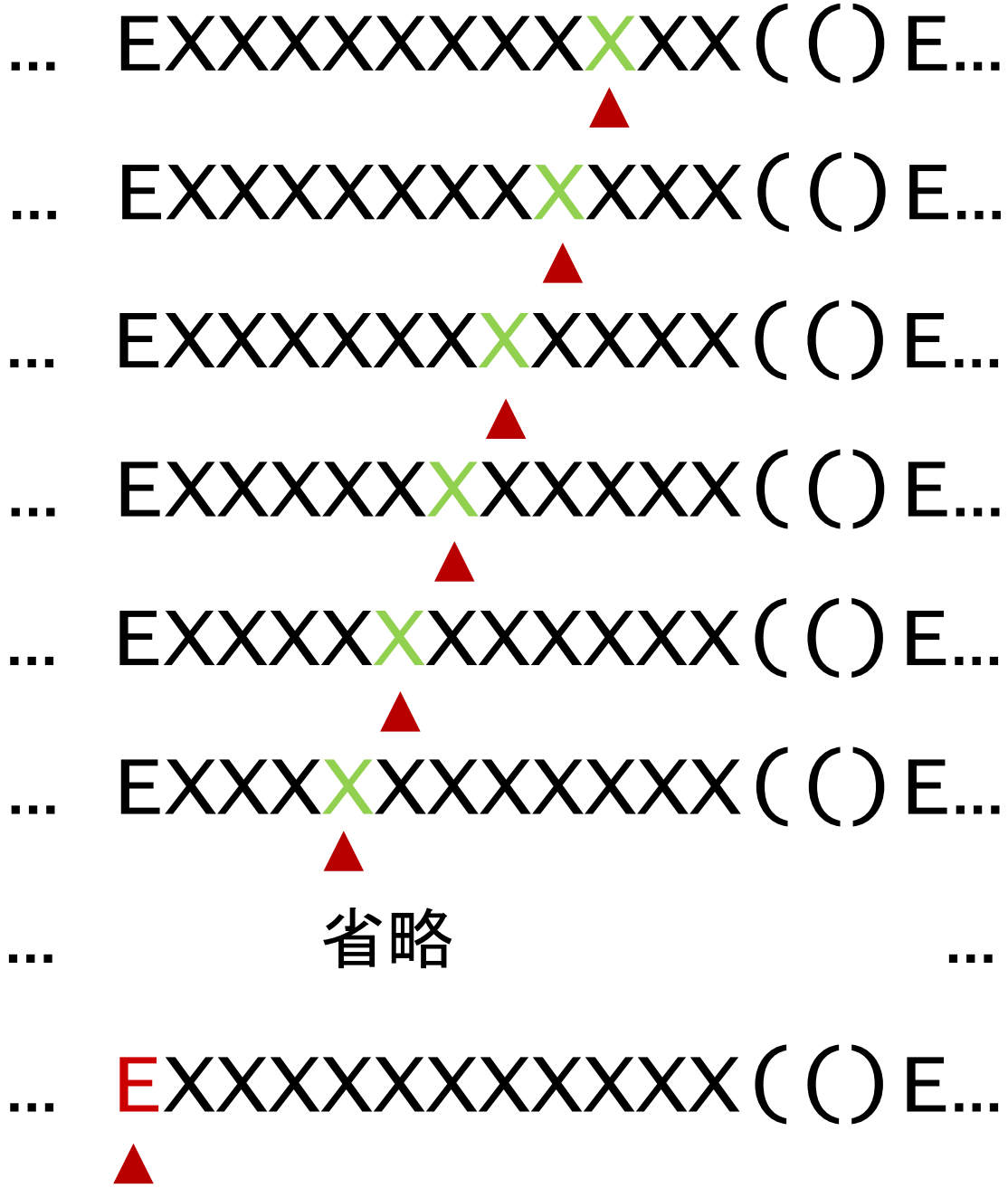
正しくない場合


... EXXXXXX(())(()E...
... EXXXXXX((X))(()E...
... EXXXXXX(XX))(()E...
... EXXXXXX(XX))(()E...
... EXXXXXX(XX))(()E...
... EXXXXXX(XX))(()E...
... EXXXXXX(XXX)(()E...
... EXXXXXX(XXX)(()E...

正しくない場合

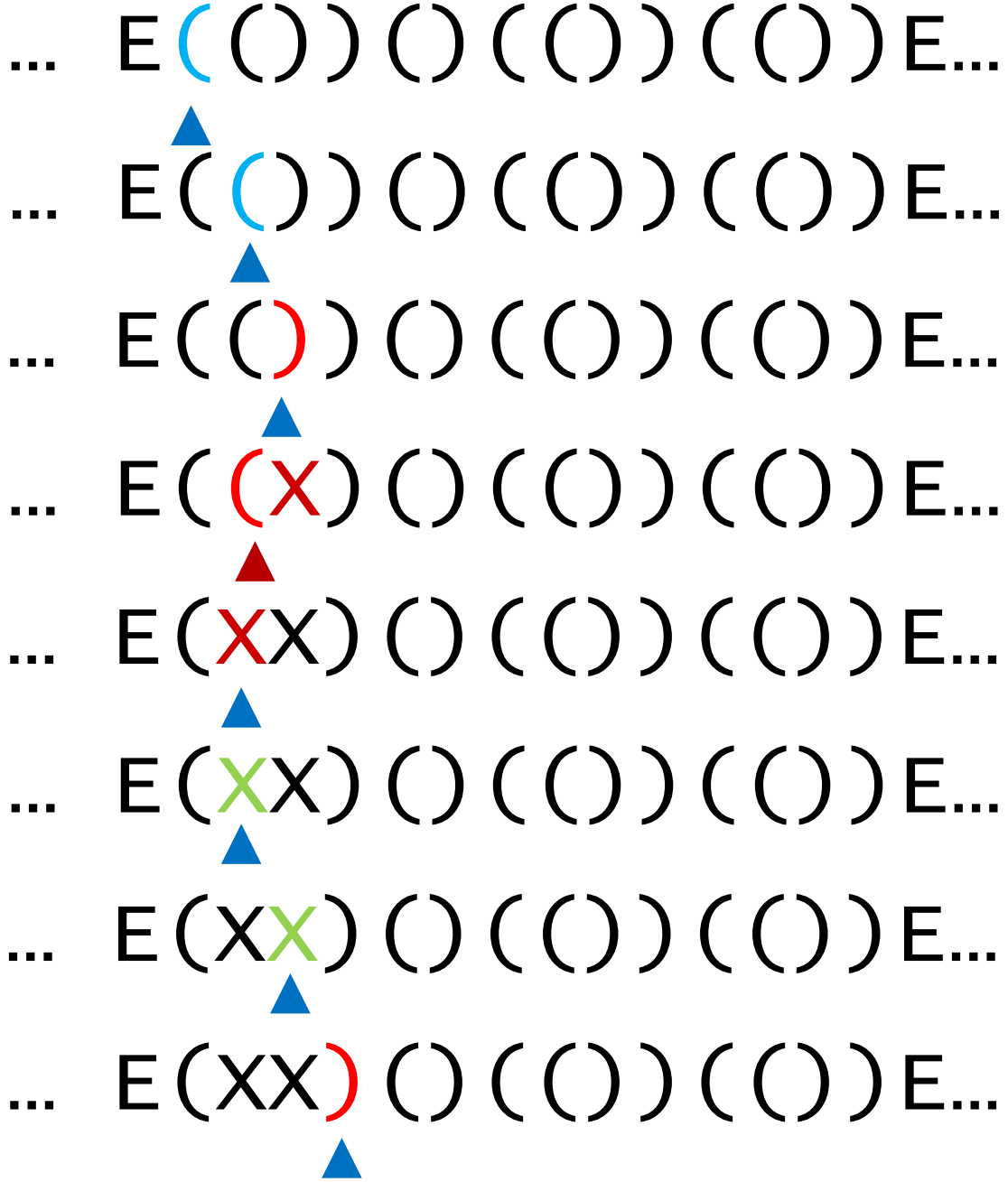

... EXXXXXX(XXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXX)(()E...
... EXXXXXXXXXXXX(()E...

正しくない場合

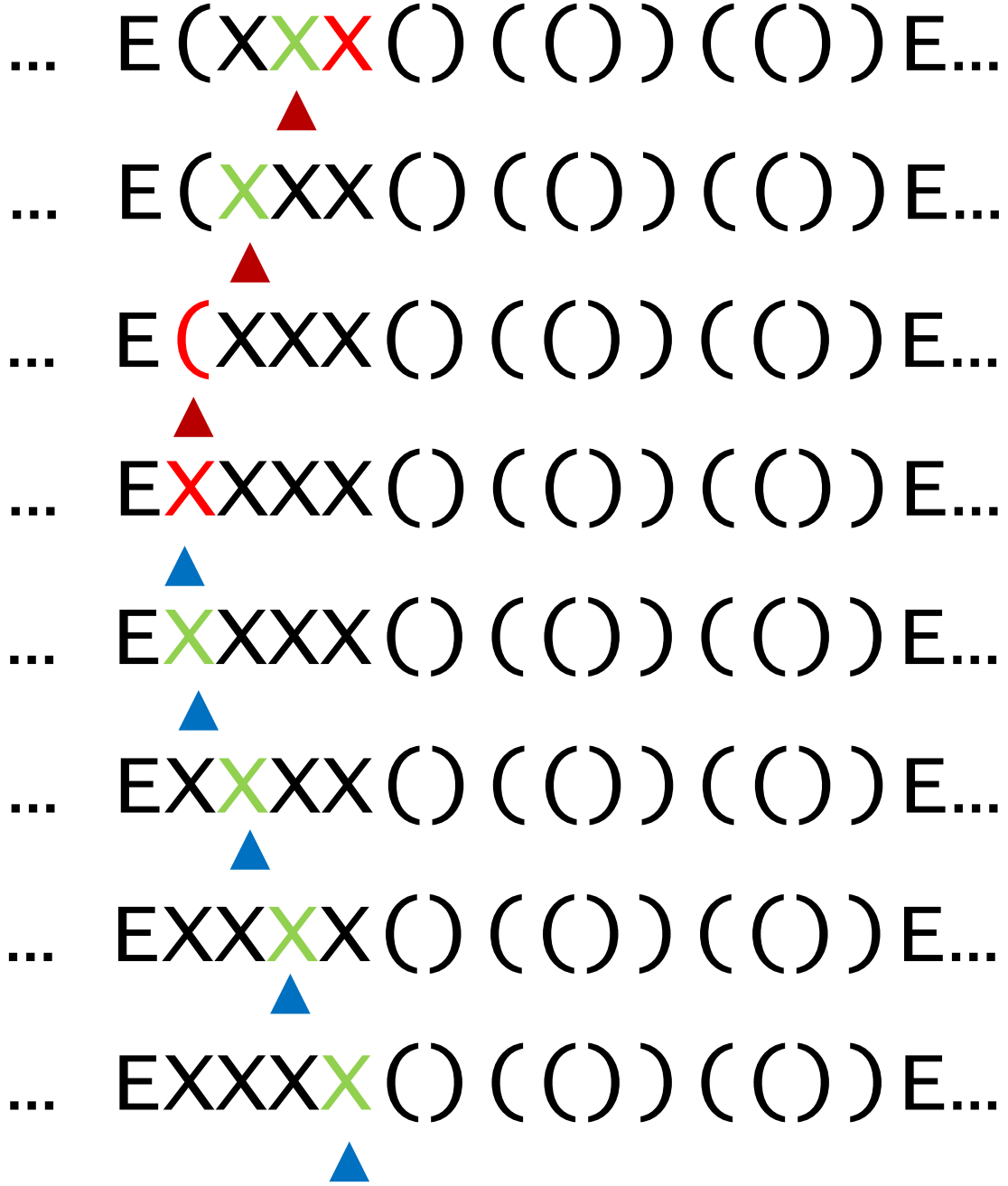

… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
… EXXXXXXXXXXXX(()E…
…
省略
…
… EXXXXXXXXXXXX(()E…

停止

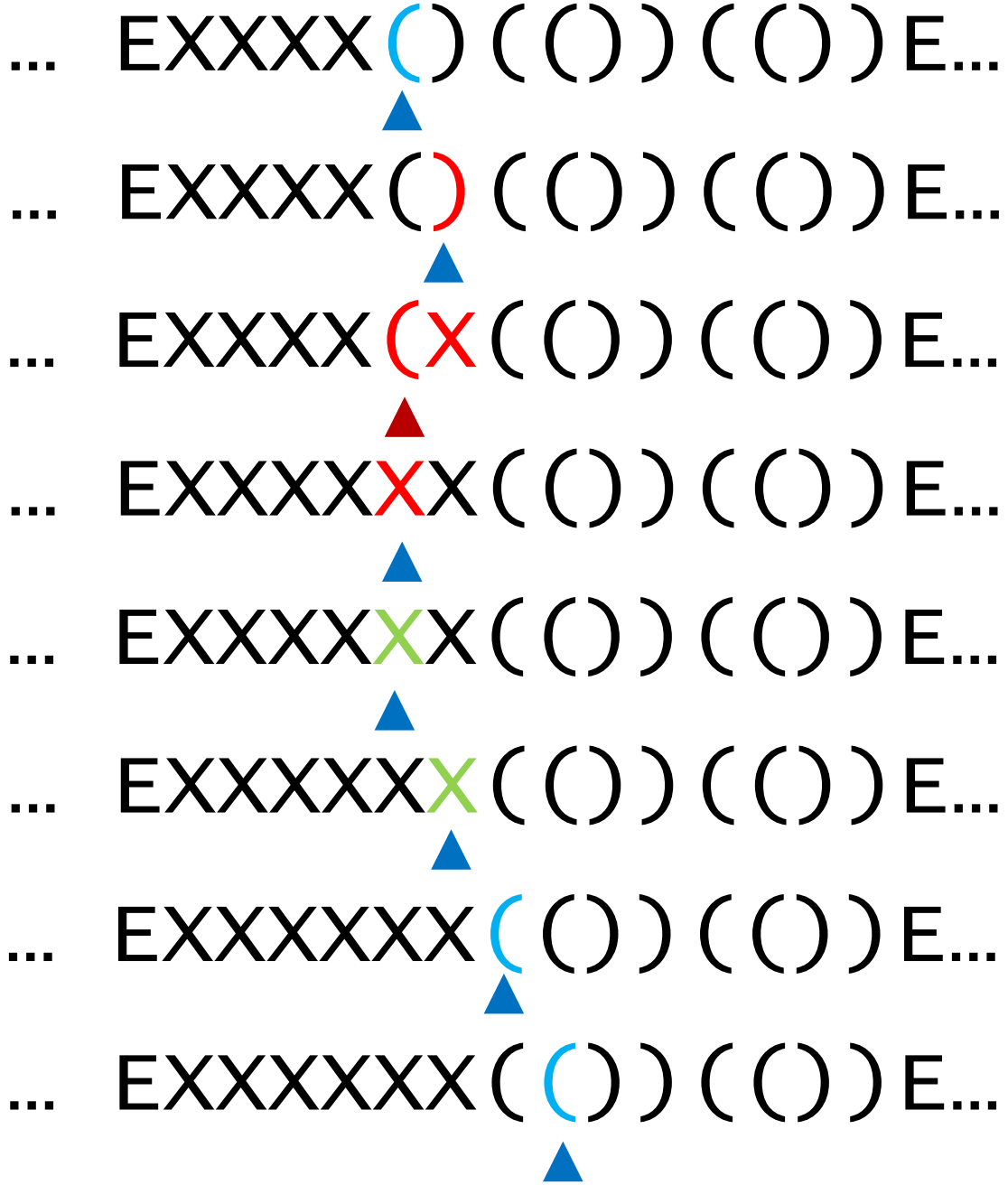
正しい場合



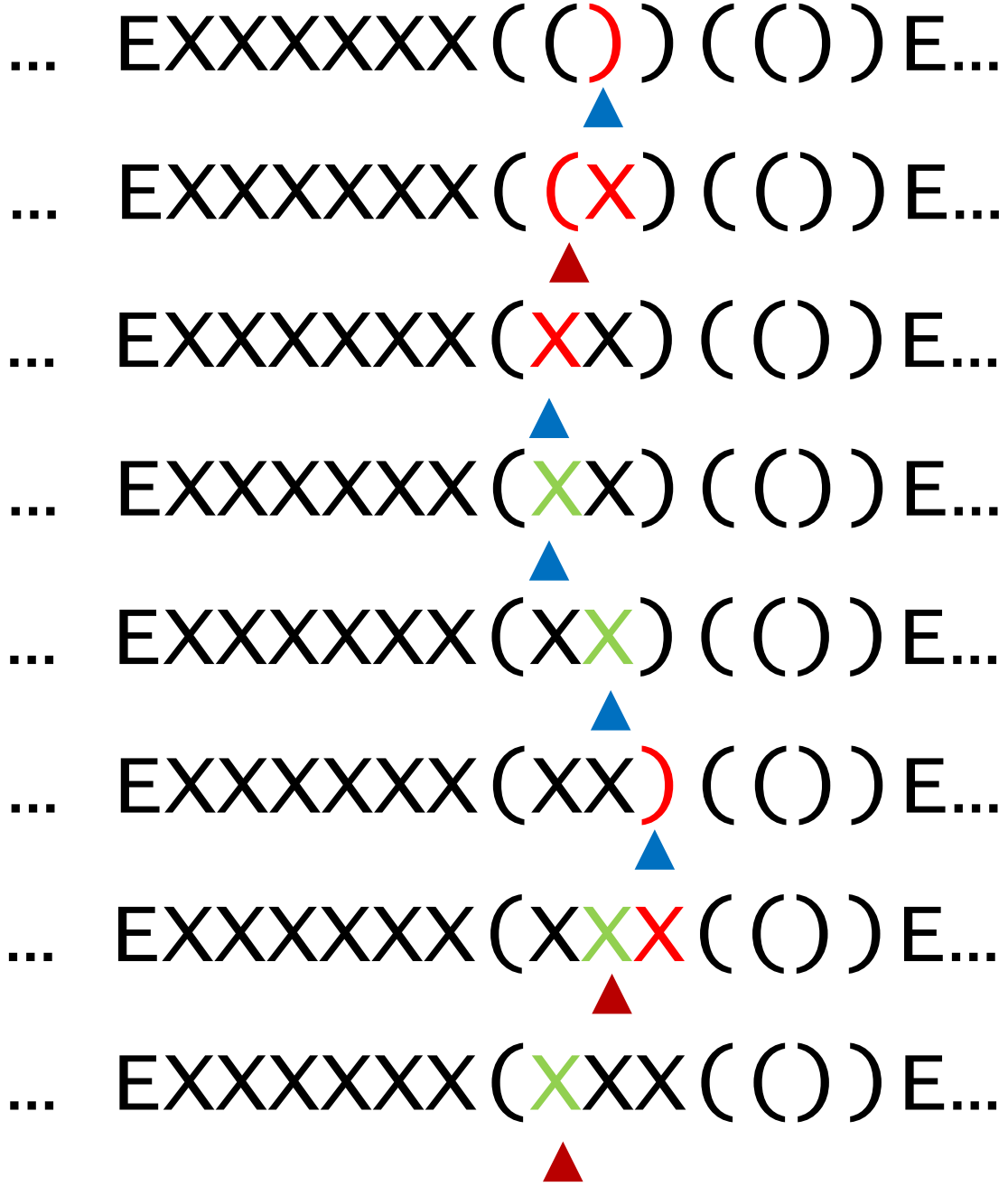
正しい場合



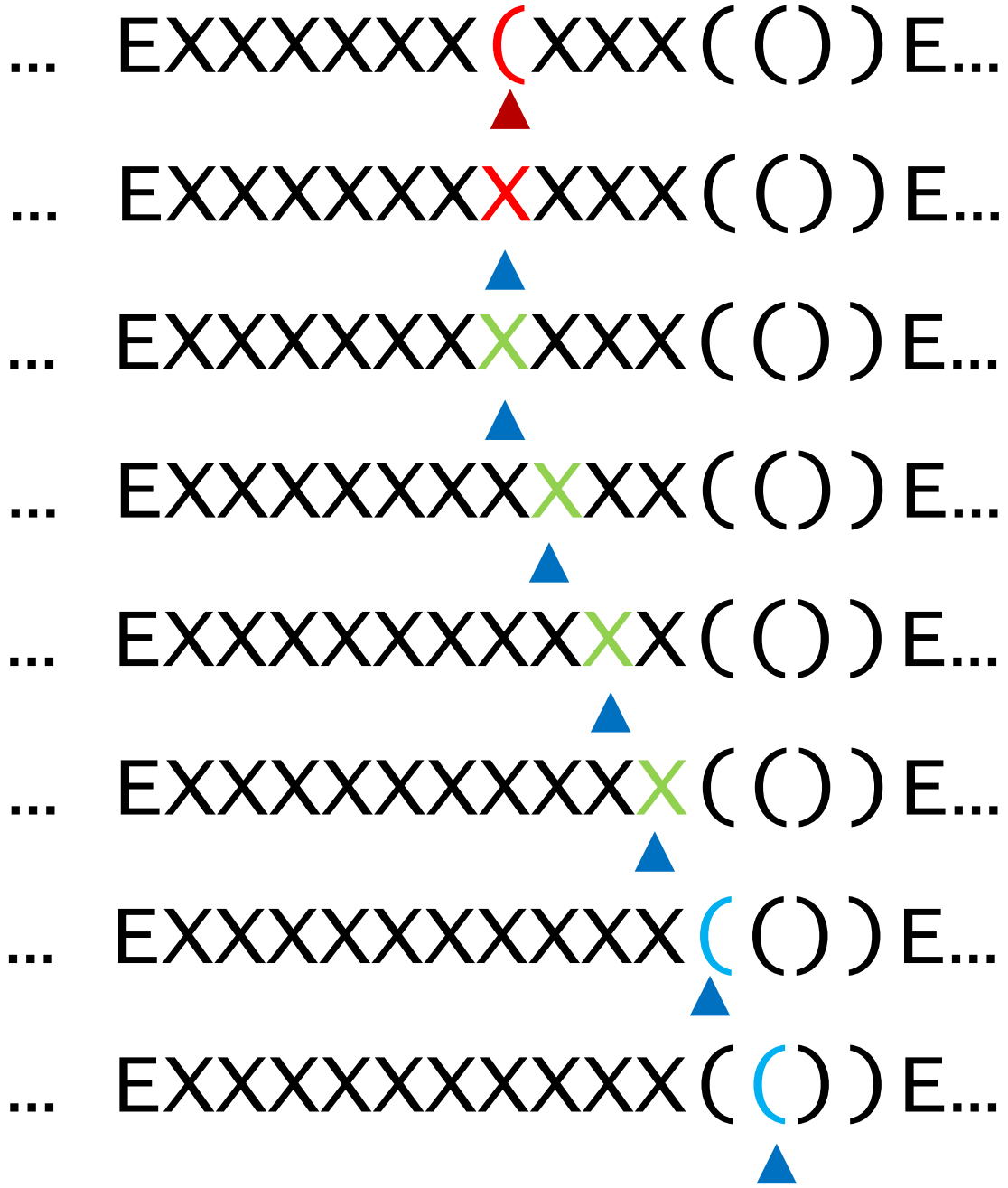
正しい場合


... EXXXX()(())(())E...
... EXXXX()(())(())E...
... EXXXX(X(())(())E...
... EXXXXXX(())(())E...
... EXXXXXX(())(())E...
... EXXXXXX(())(())E...
... EXXXXXX(())(())E...
... EXXXXXX(())(())E...

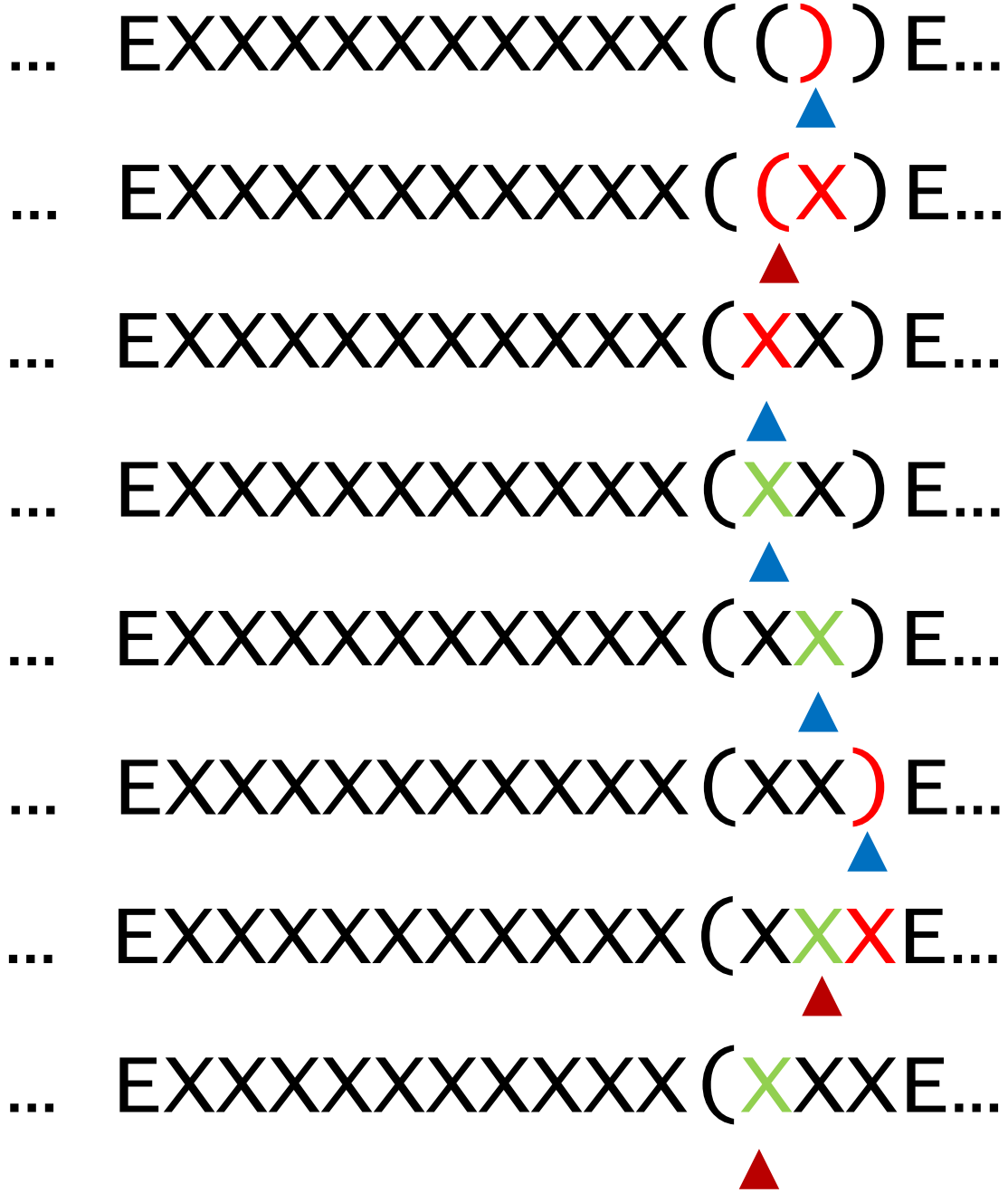
正しい場合


… EXXXXXX(())(())E…
… EXXXXXX((X)(())E…
… EXXXXXX(XX)(())E…
… EXXXXXX(XX)(())E…
… EXXXXXX(XX)(())E…
… EXXXXXX(XX)(())E…
… EXXXXXX(XXX(())E…
… EXXXXXX(XXX(())E…

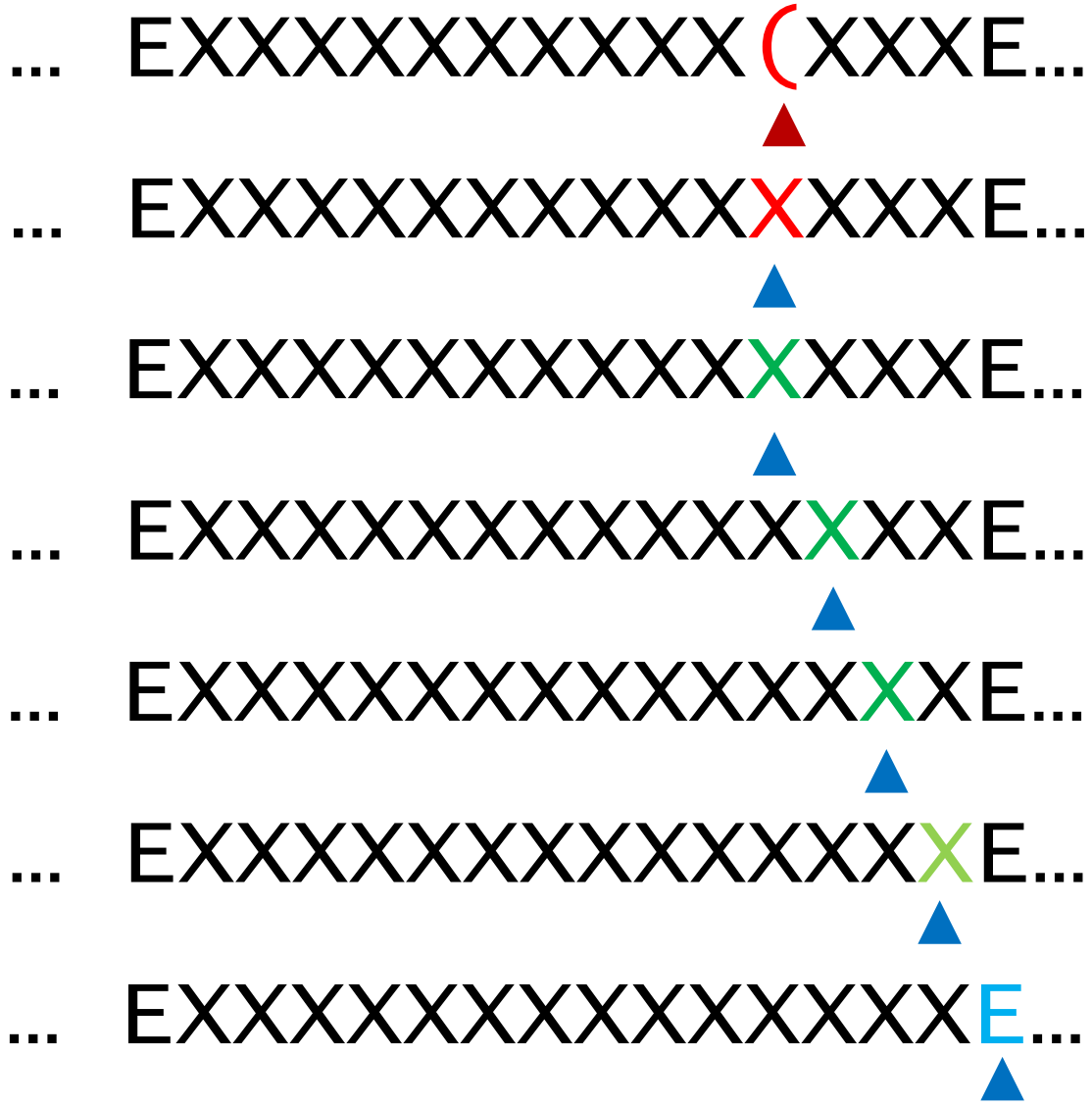
正しい場合


... EXXXXXX(XXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...
... EXXXXXXXXXX(())E...

正しい場合


... EXXXXXXXXXXX(())E...
... EXXXXXXXXXXX((X)E...
... EXXXXXXXXXXX(XX)E...
... EXXXXXXXXXXX(XX)E...
... EXXXXXXXXXXX(XX)E...
... EXXXXXXXXXXX(XX)E...
... EXXXXXXXXXXX(XXXE...
... EXXXXXXXXXXX(XXXE...

正しい場合



明らかにE(((Eもこの状態になる。
先頭のEまで戻って、(が含まれなければいい。

チューリング・マシンと
コンピュータのプログラム

例5:　無限ループ

問題5: 無限にループして停止しない例を考えよ

例1

BとBにはさまれた0,1の列が入力として与えられているとしよう。

1. 二つの状態 RightとLeftがあるとする。
2. 初期状態は、Right。
3. 状態がRightの時、Bを見つけるまでヘッドを右に移動する。
4. 状態がRightでBを見つけたら、ヘッドを一つ左に移動し、状態をLeftに変える。
5. 状態がLeftの時、Bを見つけるまでヘッドを左に移動する。
6. 状態がLeftで Bを見つけたら、ヘッドを一つ右に移動して、状態をRightに変える。

この例では、入力がB0BでもB1Bでも、無限ループで停止しない。

例6:　Goto L1, Goto L2

問題6: ヘッドを特定の位置に移動する例を考えよ

BとBにはさまれた文字列の中に、特別な文字L1とL2がそれぞれ一つづつあるとしよう。

1. GotoL1とGotoL2という状態を追加する。(他の状態の場合の振る舞いは省略する)
2. 状態がGotoL1になったら、ヘッドを先頭に移動して、文字L1を見つけるまでヘッドを右に移動する。Liが見つかったら、ヘッドを右に一つ進めて、状態を初期状態Initにする。
3. 同様に、状態がGotoL2になったら、ヘッドを先頭に移動して、文字L2を見つけるまでヘッドを右に移動する。L2が見つかったら、ヘッドを右に一つ進めて、状態を初期状態Initにする。

Gotoについて次の問題を考えよ。

1. なぜ、Gotoで、ヘッドを先頭に戻すのか？
   - この条件がないと、Gotoでジャンプしてからの処理で、再び状態が、GotoL$_i$に変わった場合、右方向に検索しているだけだと、L$_i$が見つからない場合があるからである。

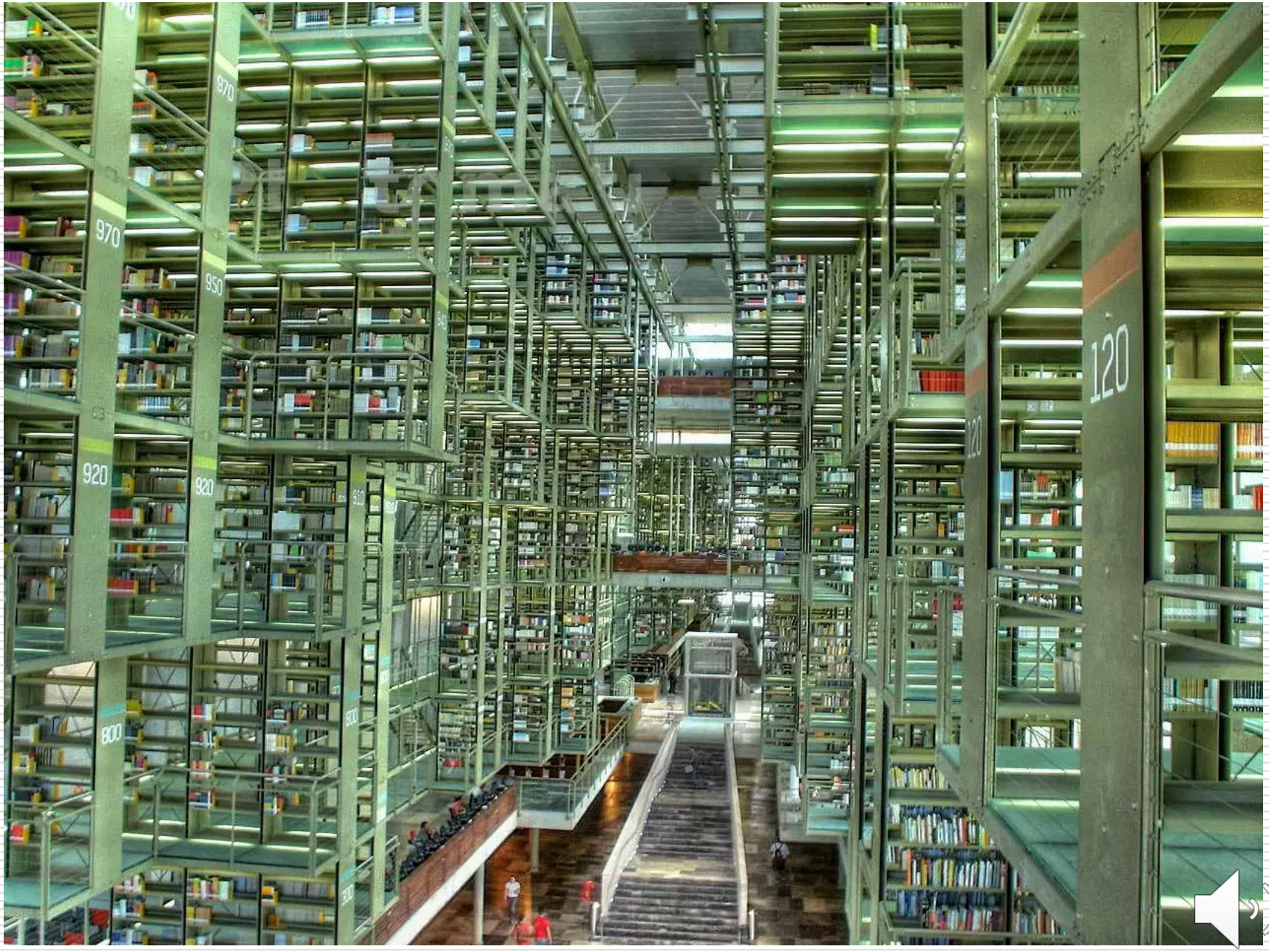
2. なぜ、Gotoで移動後、状態をInitにするのか？
   - 必ずしもInitにする必要はない。再帰呼び出しを使うなら、Initではなく、再帰用のEnvに状態を変えればいい。

3. Gotoを使って、無限ループを書け。
   - GotoL1でジャンプしてからの処理の最後に、状態をGotoL1に変えるコードを置けばいいい。

例7:　条件分岐

問題7: IF文に相当する処理を書け。

1. 状態CondTrueと、状態をCondFalseを追加する。
2. 特定のセルの値が1の時、状態をCondTrueに変え、
   そのセルの値が0の時、状態をCondFalseに変える。
3. 状態がCondTrueの時の振る舞いは、GotoL1と同じ振る舞いをするようにし、状態がCondFalseの時の振る舞いは、GotoL2と同じ振る舞いをするようにすればいい。


970
950
920
800
120


Turing Machineと
計算可能な数

Agenda
Turing Machineと計算可能な数

- チューリング・マシンと言語理論
- チューリング・マシンで計算できる数
- チューリング・マシンと停止問題
- 計算可能な自然数上の関数の個数
  - 乱数列
  - Conwayの「自由意志定理」
- 計算可能だが、とても「計算が難しい」ものがあること
  - アッカーマン関数
  - Busy Beaver 問題
- 複雑性理論へ – nと2^n

チューリング・マシンと言語理論

チューリング・マシンで
文字列の並びをチェックする

- チューリング・マシンで、与えられた文字列が、ある条件を満たしているかのチェックができる。

- 例えば、全ての文字列が '1' で出来ているかのチェックは、次のようなマシンで簡単に可能である。

1. 状態Initと、状態 One, AllOne, NotAllOneを用意する。
2. 文字'1'に会ったら状態をOneに変え、ヘッドを右に進める。
3. 文字'0'に会ったら状態をNotAllOneに変え、停止する。
4. 文字列の終了のマークに出会ったら、状態をAllOneに変え、停止する。

これまで見た例のいくつかも、文字列パターンのチェックである。

- 例1： 文字列の長さを求める
- 例2： 文字列中の'1'の数の偶奇を求める
- 例4： 括弧のバランスのチェック

あるチューリング・マシンが、入力されたある文字列をチェックして、それがあるパターンに従っていると判断して停止する時、チューリング・マシンM は文字列sを受理する(accept)という。

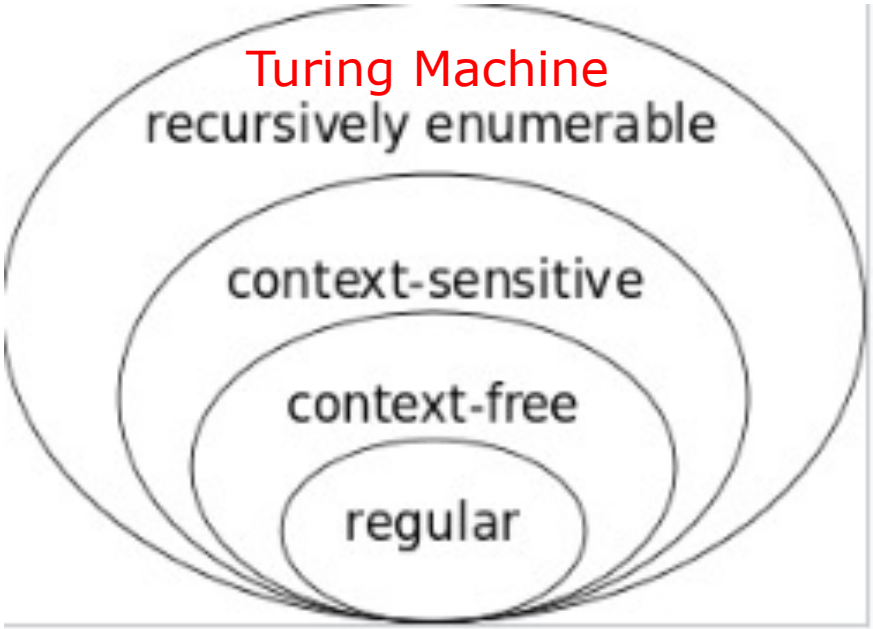
これらを少し、一般化してみよう。

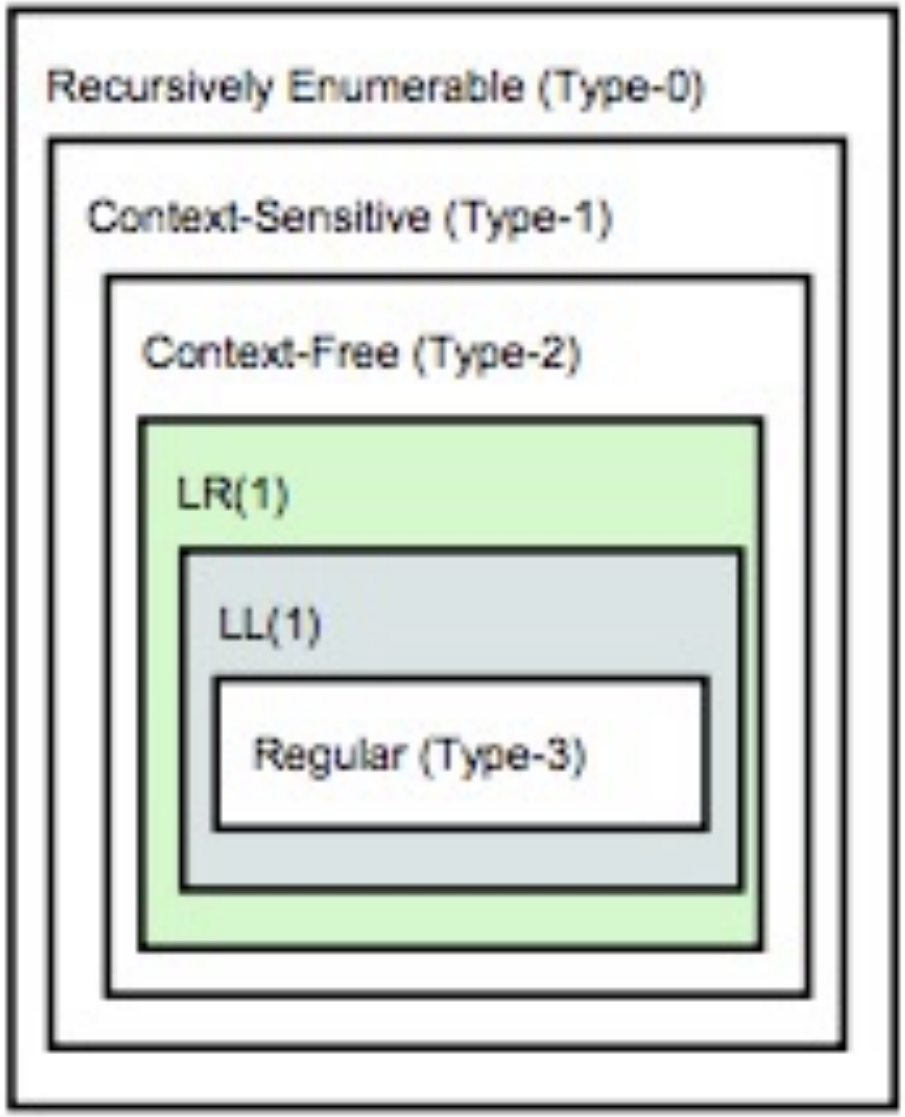
文字列パターンの「言語」としての形式化

- ある言語Lの「文」Sは、その言語の「アルファベット」の並びである。アルファベットをΣとすると、$S \subset \Sigma^*$ である。
- ただし、その言語のアルファベットの並び Σ^* が、すべて「文」になるわけではない。その言語の「文法」が、あるアルファベットの並びを「文」にする。
- あるチューリング・マシンMが、言語Lの正しい「文」 全てを「受理」する時、チューリング・マシン M は、言語Lの「文法」を記述しているという。

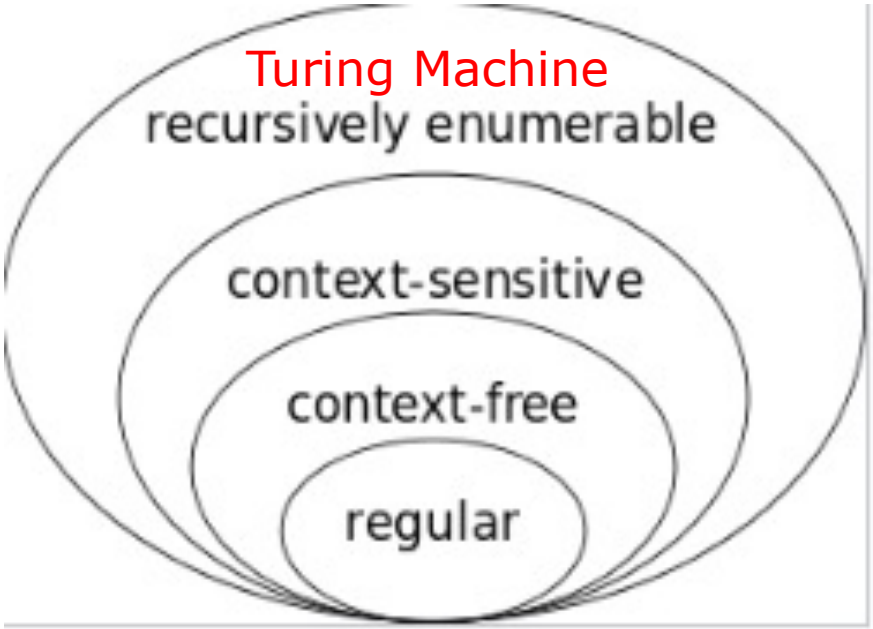
- 例えば、Lのアルファベットが、$\Sigma = \{ \text{`a'}, \text{`b'} \}$の二文字で、Lに属する文が、全て "$a^n b^n$"の形をしている時、Lの全ての文を受理するチューリング・マシンを構成できる。

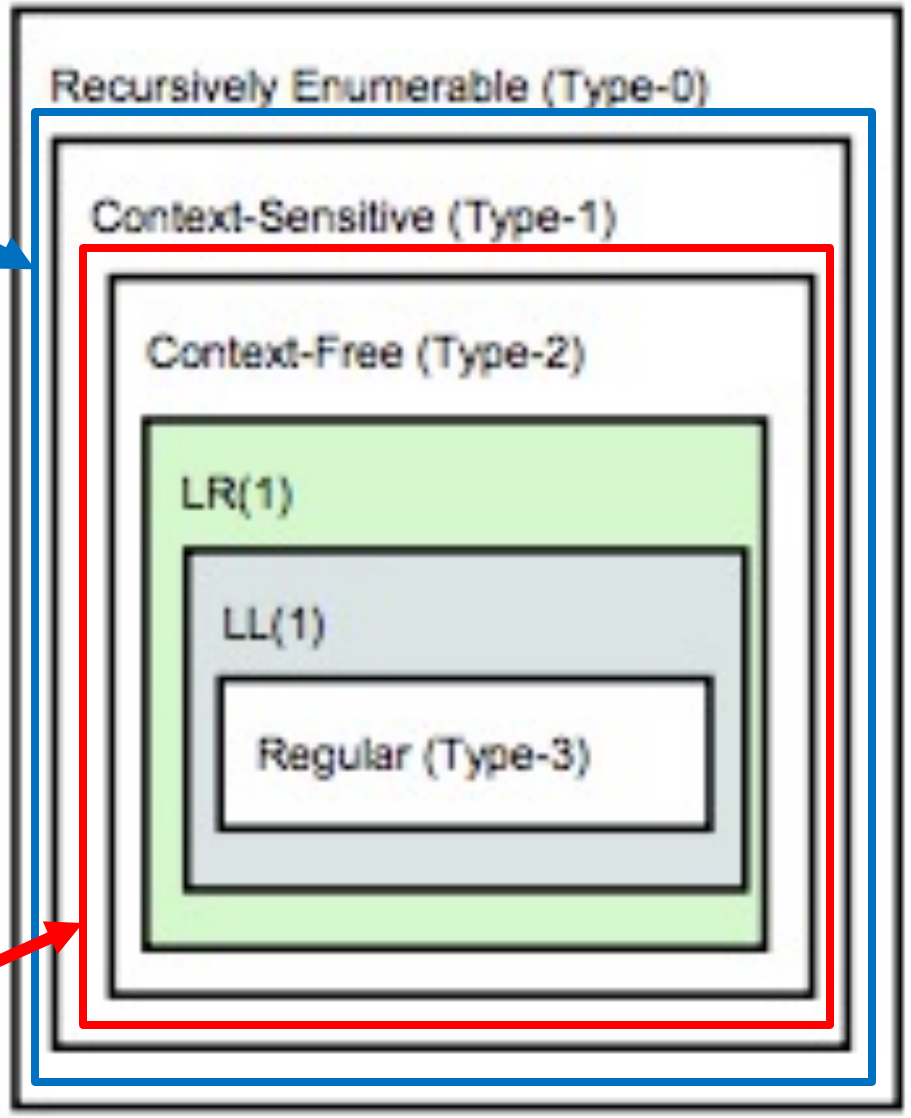
Chomsky Hierarchy





Recursive Language
"Merge" is recursive

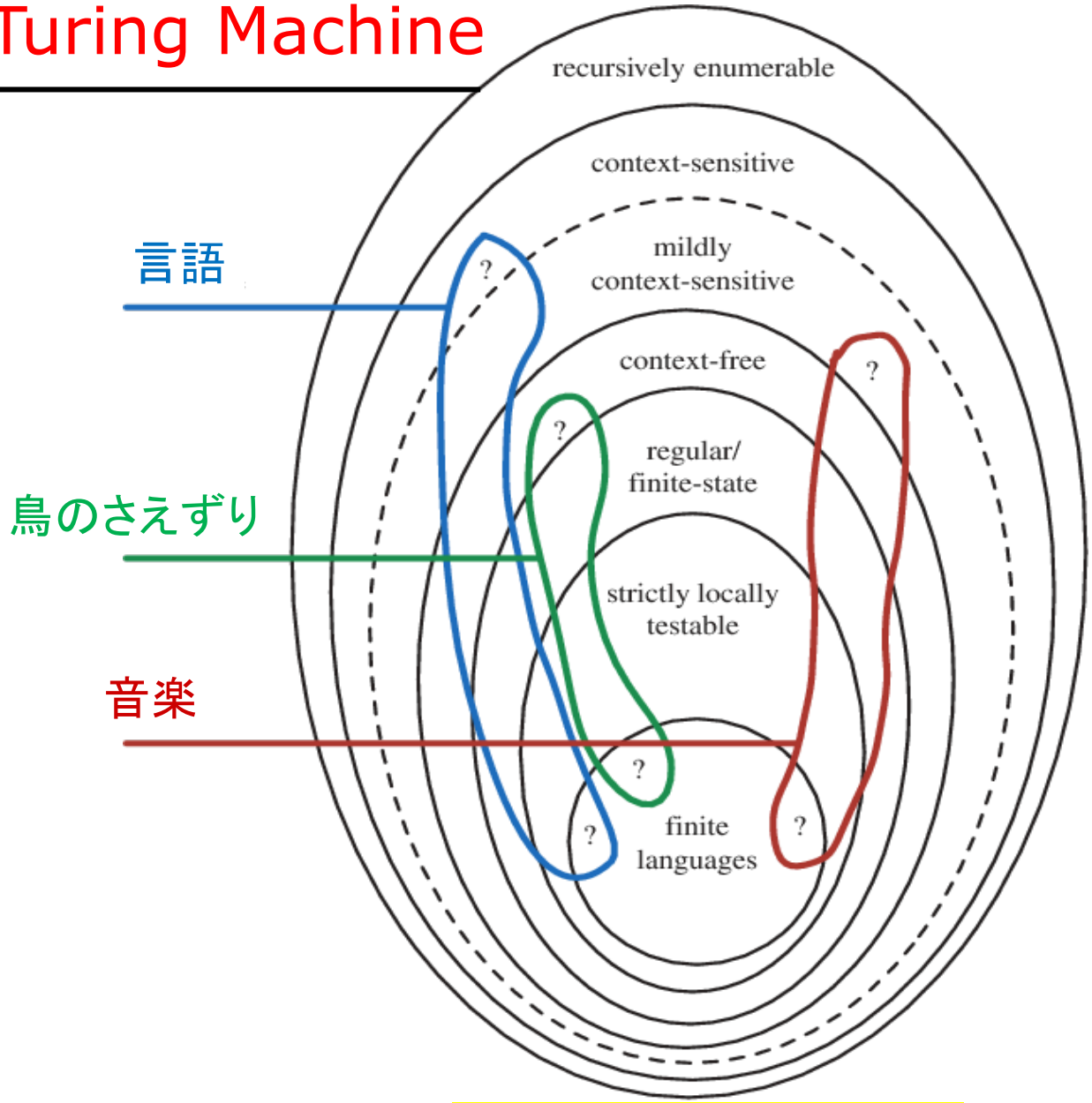




自然言語？

Mildly Context Sensitive Languages

Turing Machine



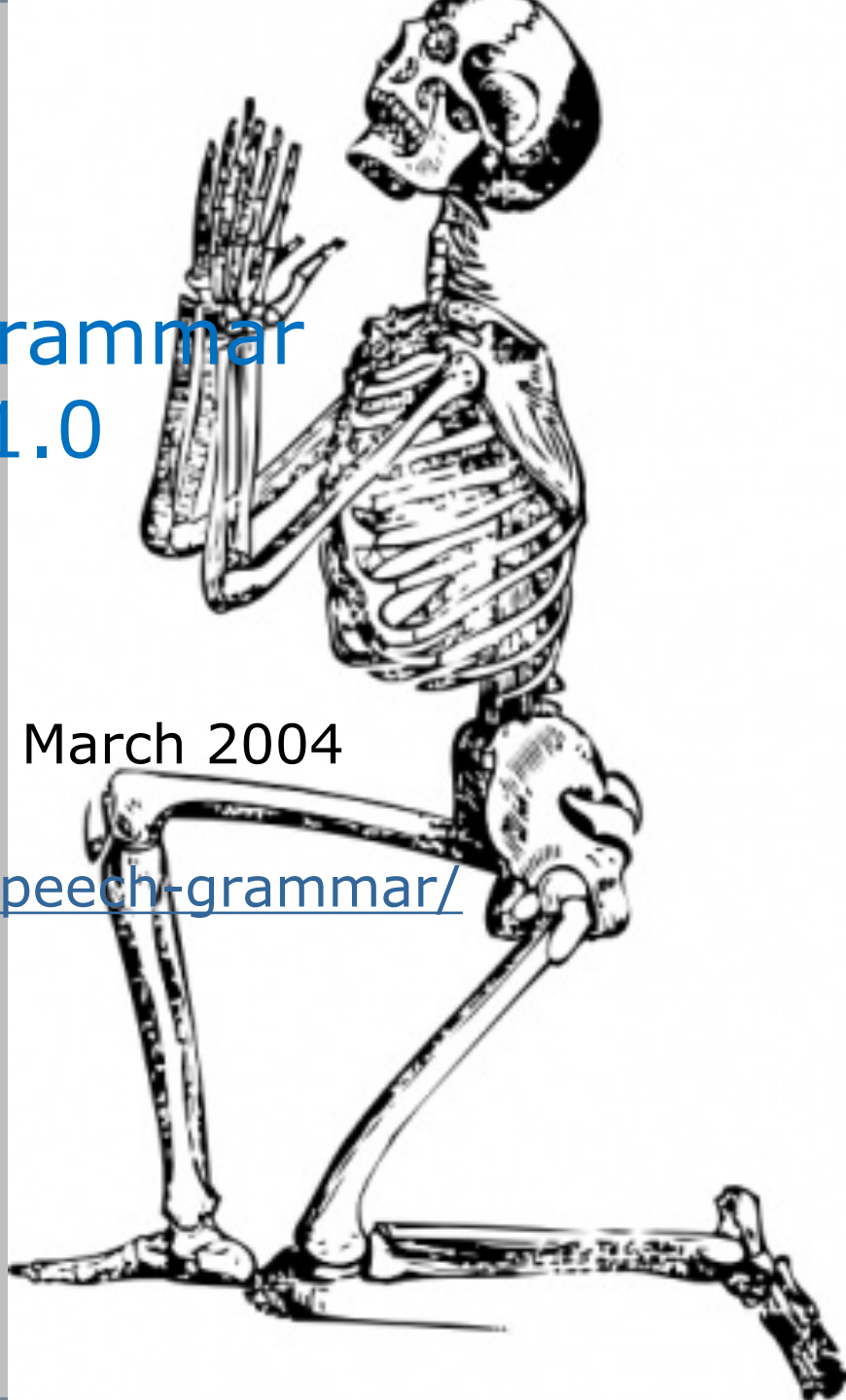
https://goo.gl/jmecp3

Speech Recognition Grammar Specification Version 1.0

W3C Recommendation 16 March 2004

https://www.w3.org/TR/speech-grammar/

ここでの「文法」は、正規文法でしかない。
それでは、コンピューター言語もコンパイラーも、作れない。



2^nの世界へ

- 先の形式化の一つとして、アルファベットが、Σ = { ‘0’, ‘1’ } からなる言語Lを考える。
- この言語Lの文sは、0と1の有限の並びである。
  $s \in \{0,1\}^* = 2^* \in 2^\omega$
- 文字列sは、文字列そのものとしても、二進表示の数としても、あるいは、あるnから 2={0,1}への関数としても解釈できる。

チューリング・マシンで計算できる数

計算可能性 ＝ 帰納性
recursive set / function

- 自然数の集合Sは、次の条件を満たすチューリング・マシンMが存在する時、「計算可能な集合」と呼ばれる。
  与えられた自然数nについて、
  - $n \in S$ なら、Mは停止して 1 を出力する。
  - $n \notin S$ なら、Mは停止して 0 を出力する。

- 自然数から自然数への関数 f は、次の条件を満たすチューリング・マシンが存在する時、「計算可能な関数」と呼ばれる。
  - 入力 n に対して、Mは停止して、出力 f(n)を返す。

こうした「計算可能な集合」「計算可能な関数」を「帰納的(recursive)な集合」「帰納的関数」と、呼ぶ。

帰納的可算
recursively enumerable

- ある自然数の集合Sに対して、チューリング・マシン Mが存在して、次の条件を満たす時、Sは「帰納的可算 recursively enumerable 」であるという。
  - Mへの入力として自然数nが与えられた時、Mが停止するのは、n ∈ S の場合だけである。

- 別の言い方をすれば、Mは、Sに属するメンバーを枚挙する能力を持つのだが、全ての入力に対してMが停止するとは限らないということである。

部分的関数
partial function

- チューリング・マシンMが停止するか否かは、Mがある関数を計算可能かに大きく関わっている。そのことを、もう少し明確にしよう。

- 次の「部分的帰納関数」の定義は、先の「帰納的可算な関数」定義と同値である。

$$\mathrm{f(x)} = \begin{cases} 1 & x \in S \\ \text{未定義 } or \text{ 停止しない} & x \notin S \end{cases}$$

帰納的と帰納的可算

- 帰納的関数は、帰納的可算な関数である。
  ただし、全ての帰納的可算な関数が、帰納的であるわけではない。なぜなら、帰納的関数は、$n \notin S$ なるnに対しても 関数が値を持つことを要求するが、帰納的加算な関数は、そうしたnについては、値が求まらないことを認める。
- そのことを、先に見た帰納的可算と同値の partial function の定義は、明示的に示している。帰納的可算なpartial functionに対して、帰納的な関数を total function と呼ぶことがある。

帰納的可算を r.e. で表すと、自然数の集合Sは、
$S \in$ r.e. かつ $\bar{S} \in$ r.e. の時に、帰納的となる。

チューリング・マシンと
停止問題

チューリング・マシンの停止問題

- チューリング・マシンの停止問題というのは、任意のチューリング・マシーンMとそれに対する任意の入力Aが与えられた時、このチューリング・マシンは停止するか否かを求めよという問題である。
- この問題を解くチューリング・マシンは存在しない。
  それを「停止問題の決定不能性」という。

決定不能性の直観的説明

- 次のような関数gが、定義可能か考えてみる。

  def g(): if halts(g): loop_forever()

- もし、halts(g)が真なら、すなわちgが停止するなら、loop_forever()が呼び出されて、gは停止しない。
- もし、 halts(g)が偽なら、すなわちgが停止しないなら、loop_forever()を呼び出さずに、すぐにgは停止する。
- いずれにしても、矛盾するので、こうした関数は定義できない。

決定不能性の証明
そうした決定可能な関数が存在すると仮定する

Turing Machine と入力を枚挙する

- Turing Machineを全て数え上げて、それをM_iと表す。
- Turing Machineに対する入力を全て数え上げて、それをA_jと表す。
- 適当なコーディング・ルールを与えれば、M_i , $A_j \in 2^*$ で表現できる。

Turing Machine Mに、入力Aを与えた時に、すべてのM,Aに対して、停止問題を決定可能なM(A)を次のように定義できると仮定しよう。

- Mが停止してある出力を返す時、M(A) ＝ 1
- Mが停止しない時、M(A) ＝ 0

そのような決定可能なM(A)が存在するとして、次のような計算表を考える

$$
\begin{array}{lccccccc}
 & A_0, & A_1, & A_2, \ldots\ldots & , A_n, & A_{n+1}, \ldots\ldots \\
M_0: & M_0(A_0), & M_0(A_1), & M_0(A_2), \ldots\ldots, & M_0(A_n), & M_0(A_{n+1}), \ldots\ldots \\
M_1: & M_1(A_0), & M_1(A_1), & M_1(A_2), \ldots\ldots, & M_1(A_n), & M_1(A_{n+1}), \ldots\ldots \\
M_2: & M_2(A_0), & M_2(A_1), & M_2(A_2), \ldots\ldots, & M_2(A_n), & M_2(A_{n+1}), \ldots\ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots \\
M_n: & M_n(A_0), & M_n(A_1), & M_n(A_2), \ldots\ldots, & M_n(A_n), & M_n(A_{n+1}), \ldots\ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots
\end{array}
$$

この計算表は、Turing Machinn M_iとそれへの入力 A_jが与えられた時、Turing Machineが停止するか否かを 1,0の値で示すものである。

計算表の対角線上の要素を取り出す

$$
\begin{array}{lccccccc}
 & A_0, & A_1, & A_2, \ldots\ldots & , A_n, & A_{n+1}, \ldots\ldots \\
M_0: & M_0(A_0), & M_0(A_1), & M_0(A_2), \ldots\ldots, & M_0(A_n), & M_0(A_{n+1}), \ldots\ldots \\
M_1: & M_1(A_0), & M_1(A_1), & M_1(A_2), \ldots\ldots, & M_1(A_n), & M_1(A_{n+1}), \ldots\ldots \\
M_2: & M_2(A_0), & M_2(A_1), & M_2(A_2), \ldots\ldots, & M_2(A_n), & M_2(A_{n+1}), \ldots\ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots \\
M_n: & M_n(A_0), & M_n(A_1), & M_n(A_2), \ldots\ldots, & M_n(A_n), & M_n(A_{n+1}), \ldots\ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots \\
\vdots & \vdots & \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots
\end{array}
$$

D $= M_0(A_0)M_1(A_1)\,M_2(A_2)\ldots\ldots\; M_n(A_n)M_{n+1}(A_{n+1})\ldots\ldots$

とする。

この計算表は矛盾を含んでいる

D = $M_0(A_0)M_1(A_1)\,M_2(A_2)\,\ldots\ldots\,M_n(A_n)M_{n+1}(A_{n+1})\,\ldots\ldots$
として、K=$\overline{D}$ を考える。
K = $\overline{M_0(A_0)M_1(A_1)\,M_2(A_2)\,\ldots\ldots\,M_n(A_n)M_{n+1}(A_{n+1})\,\ldots\ldots}$

Kが先の計算表に含まれているとしよう。
すなわち、ある i について、M_i = K としよう。
この時、$M_i(A_i)$ = K(A_i) となる。
ところで、Kの定義 K=$\overline{D}$ から、K(A_i) = $\overline{M_i(A_i)}$
$M_i(A_i)$ = K(A_i) = $\overline{M_i(A_i)}$　　これは矛盾である。
よって、Kは先の計算表には含まれない。

自然数と実数の一対一対応が存在しないことの別証明

r_i を0と1の間の実数の二進表示とする。

0と1との間の実数全てを、次のように自然数と一対一に対応付けることが出来たとしよう。

$$0 : r_0 \ , \ 1 : r_1 \ , \ 2 : r_2 \ , \ \dots \ , \ n : r_n \ , \ \dots$$

r_i は、$r_i \in 2^\omega$という関数としても考えることができるので、次のような 0と1の並び Dを考える。

$r_n(r_n)$ は、関数r_nに引数r_nを与えたものである。0か1が返る。

$$D = r_0(r_0)r_1(r_1)r_2(r_2) \dots r_n(r_n) \dots$$

今度は、このビット列を反転してKとする。$\overline{0}$=1, $\overline{1}$=0, $\overline{101}$=010

$$K=\overline{D}=\overline{r_0(r_0)r_1(r_1)r_2(r_2) \dots r_n(r_n) \dots}$$

自然数と実数の一対一対応が存在しないことの別証明

DもKも、0,1の無限列で、0と1の間の実数の二進表示と見なすことができる。同時に、DもKも、D, K $\in 2^{\omega}$ という関数とみなすことができる。

Kが先の対応リストに含まれているとしよう。すなわち、ある i について、r_i = K としよう。この時、$r_i(r_i)$ = K(r_i) となる。

ところで、Kの定義から、K(r_i) = $\overline{r_i(r_i)}$

$r_i(r_i)$ = K(r_i) = $\overline{r_i(r_i)}$　　これは矛盾である。

計算可能な自然数上の関数の個数

計算可能な自然数上の関数の個数

- チューリング・マシンで定義可能な自然数上の関数は、帰納的可算である。
- チューリング・マシンで計算可能な関数すなわち帰納的な関数は、さらにそのサブセット（その補集合も帰納的可算でなければならない）なので、たかだか可算個しか存在しない。
- 一方、自然数から自然数への全ての関数は、2^{ω}で表示できるので、実数と同じ個数だけ存在する。
- すなわち、自然数から自然数への関数は、計算可能な自然数から自然数への関数より、はるかに多いのだ！
- 別の言い方をすれば、自然数から自然数の関数の大部分は、計算不能なのである。

0と1の「ランダム」な並び

- 計算不能な自然数から自然数の関数 rというと、なにか不思議な関数が存在していると考えるかもしれない。でも、そうではない。それは、次のようにしてわかる。
- 二進表示で $r \in 2^{\omega}$ とすると、それは、0と1の無限列である。ここで、0と1が「ランダム」に登場する 0と1の無限列を考える。「ランダム」というのが、「予測ができない」「ルールが存在しない」という意味だとすれば、この0と1の「ランダム」な並びが、計算不能な自然数から自然数の関数 だと考えればいい。
- そうした並びが、非可算個存在しうるのも、直観的には納得できる。
- 計算可能な関数は、きちんと定義できるのだが、逆に、「ランダムさ」を、きちんと定義するのは難しいのだ。

Conwayの「自由意志定理」

John. Conway, Simon Kochen, 2006年
“The Free Will Theorem”
https://arxiv.org/pdf/quant-ph/0604079.pdf

Abstract

- On the basis of three physical axioms, we prove that if the choice of a particular type of spin 1 experiment is not a function of the information accessible to the experimenters, then its outcome is equally not a function of the information accessible to the particles. We show that this result is robust, and deduce that neither hidden variable theories nor mechanisms of the GRW type for wave function collapse can be made relativistic. We also establish the consistency of our axioms and discuss the philosophical implications.

1. Introduction

- Do we really have free will, or, as a few determined folk maintain, is it all an illusion? We don’t know, but will prove in this paper that if indeed there exist any experimenters with a modicum of free will, then elementary particles must have their own share of this valuable commodity.

- **Fin:** There is a maximal speed for propagation of information (not necessarily the speed of light). This assumption rests upon causality.
- **Spin:** The squared spin component of certain elementary particles of spin one, taken in three orthogonal directions, will be a permutation of (1,1,0).
- **Twin:** It is possible to "entangle" two elementary particles and separate them by a significant distance, so that they have the same squared spin results if measured in parallel directions. This is a consequence of quantum entanglement, but full entanglement is not necessary for the *twin* axiom to hold (entanglement is sufficient but not necessary)

計算可能だが、
とても「計算が難しい」ものがあること

一般帰納関数として定義された
原理的には計算可能な「アッカーマン関数」

一般帰納関数の計算可能性
アッカーマン関数

次のような関数を考える。

$$\mathrm{Ack}(m,n) = \begin{cases} n+1, & \text{if } m = 0 \\ \mathrm{Ack}(m-1,1), & \text{if } n = 0 \\ \mathrm{Ack}(m-1,\mathrm{Ack}(m,n-1)), & \text{otherwise} \end{cases}$$

この関数は、一般帰納関数として定義されているので、チャーチ＝チューリングのテーゼの条件を満たしている。

ただ、m,nの値が、少しでも大きくなると、この関数の値は、爆発的に増大して、普通のコンピュータでは、事実上、計算が不可能になる。

次ページに、Ack(m,n)の値を示す。

A(*m*, *n*) の値

*m**n*	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2=2+(n+3)-3$
2	3	5	7	9	11	$2n+3=2\times(n+3)-3$
3	5	13	29	61	125	$2^{n+3}-3$
4	13	65533	$2^{65536}-3$	$2^{2^{65536}}-3$	A(3, A(4, 3)) = $2^{2^{2^{65536}}}-3$	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3\text{ twos}}-3$
5	65533	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{65536\text{ twos}}-3$	A(4, A(5, 1))	A(4, A(5, 2))	A(4, A(5, 3))	$\mathrm{A}(4,\mathrm{A}(5,n-1))$
6	A(5, 1)	A(5, A(6, 0))	A(5, A(6, 1))	A(5, *A*(6, 2))	A(5, A(6, 3))	$\mathrm{A}(5,\mathrm{A}(6,n-1))$

我々に手の届かない有限について
Busy Beaver 問題

Busy Beaver 問題

「忙しいビーバー」問題は、状態の数がN個のTuringマシンを全部考えて、全て0で埋まっているテープから初めて、一番長いステップで停止するTuringマシンとそのステップ数を求めよという問題である。その最大値をBB(N)で表すことにすると、次の結果が知られている。

BB(1) = 1.
BB(2) = 6.
BB(3) = 21
BB(4) = 107

実は、まだ、誰も、BB(5)の値を知らない。

BB(5) ≥ 47,176,870

というのはわかっている。（誰かが、状態の数が5つで、47,176,870ステップで停止するTuringマシンを作ったということ。ただ、それが「最大」だということはわかっていない。）

そんなものかと思われるかもしれないが、雲行きが怪しくなるのは、BB(6)あたりから。

BB(6) ≥ 7.412 x (10の36,534乗)

だという。36,534桁の数字になる。先のBaezのblogに、「具体的に」この数字が紹介されている。延々と数字が続く。3万6千桁以上の数字が。

よくそんなことわかるなと思われるかもしれないが、それは、先にも述べたように、実際にそういう6状態のTuringマシンを構成して見せたからだ。実は、先に紹介した、6個の状態と12個の命令からなるTuringマシンが、そういう動き方をするのだ。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC

「プログラム自体は、単純である。」と先に書いたのだが、正しくないかもしれない。このプログラムは、えんえんと走り続け、そして、最後に停止する。(停止すること自体、不思議な気がする)

741207853509495610174172561144604969718281611695298008925669010951656624280328485493565509745496832570980660176344529980240910175257246046044979228025757711518548052087650589935156483217413541197777965279255493532447649712935848974978461539867784215790591584199376184970716056712502662159444630412079992352830139286757106976976266378010157256661988247506945421793112446656331449750558811894710601772365595995396507670767065001451170294752766860167506529522954129044871107849518263169509747269711158732776867610634089559834790714490552734463125404673708090108604513212129769190196259350728934650321231429040253205457633515626107868771760119916923828746803714584592161274169396551793596257979821625066031449422781829328977925461473293508048670145466821939225145869908038228128045266110256571782631958926898525694689966695874227519616911181790608398001974214915398771591696883364753477480074875777666112880815431005997890623859440699663891591940342058445345135951600168916065895276541430702668840252531350653890897051982632630385938083660639947985722351182179370081120817877269116559398341767052162655917201202433328300323752878230642831971805072395297353229551731048371021811597619301167315826868031 1

735322955173104837102181159761930116731582686803119630571008041911930477971579672785029529559439797294500432643483372677378872480519292318129075141594300170423748519482724090149197762230095180896645720799274350771114821420869806320389838466348644400634378985820511533007233636175063664244348363429381716865277675927173868435130844307605385977884978545703928873633762169439444600268593765042490439747070469396499307353236961551408770635423051623551580205020464330288028606243330668268170811900863960680544921270550820866591183165171362199062268022146340355100698032539208500321737935916572824167109243921796327708886988064622392869223657202340493532622931976010973933691955155585614971701393657083510829337138019755841416703048425473095781543947284695305578910483032968871052034458997996570053793215656951656746253672334104502853352759013244455004163729329181785691615066366432555395455966924963740865923479058518089001727259875665774273483385120744001467195339376792213348011167418166728360082829942361956450241322000295203110123701834947807654204715508724845292827346100146348547365089126533446859392192538154695164187041834978200787084142435296024 6

816219279435128009691488333367252441723619461885472096381488087746264915498258124361228833219320352241878334479833109122807808425070272194370533803098155762074363594124056079914285825861353250006004503404457075561484235380160575513851472863744453819991270653752636827482388619627545586769702982563550215791905943470016781247941475207377805457107252380926307057894825122170537840420055751812318342976374391628221317569903581457033268409573939140317537929519452225728328543766673545899812218722084639419217330239037159748031355083246976463886069438573523920879420538715366934472880272772745254215764827226580772102826496399117753878844601174815425740209860471059749736357781622490624052946817662800124337027642430572009172724680494845807607875336391296355953749364637564991523417213639553068550050631478405859742434139260653244354541439306503595217538645638222669677018885647203581909266795843083183075450785277713783211861708706612681436736614104408791294005647913540430281084317983076118608172715678564098233869131431441387859096996327035057252704630665023999285518292849124642753224570810976792933497725693910894339658778178382786680971310533947980194252766851530607523746692117596241149131822801952

28443629054406029354014078168448839468543109777746497134194328220740395976456632163669113880556744440040338242074160211898209536897949768268405300638550209609958621490671271332429239552166892883811184405888820990463604425076520669497073794980146362709477533118591401481473656166664409698471099509772674271268524194763314787756784416422586929186303999309479972891692726739231712531500339659100715122651178203821487177649041575923087270542299624201754570706993341240356064699636293209512874013786250682473887757931001901807158800515178567563191206326463879171290239696789072427830303321073398269847363456290199268793655334873976194500237322039977440978878227032599708324913637134795947392057672257001889829885987903466227757446048410092755426060057477348984785786907788507119614119876333605601699259296191798210522981667181477607820683943238312997333502226273311447560030346344769138042632040645897100672747110856632877598317707068109285992387448288963033783848284343008603095759504211313687275146815571999153003835709371811628295885386861489714678254967080333475258187514533923483088945298272830364477058058993426870144942681269232766983593731414821

4467492875119988039620924341039458961770757345685
6454301520275813300267434717502921849792945757378 6
6546765128985326270025306439142280125170385670430 4
3993367412997435263933332827902185399873234049339 1
5243986633960766937577765446089358460929132081948 2
3545029493121851964625769147302478463529246280565 4
6056581254571018956482544451653310465454962630777 4
1375950179168158540681999239199541083222930573503 1
7974307367947840165992935953268841291762992863264 6
2353158681102294807472460144080798083883059422402 0
7030904215718776078177940150483268879448134664598 9
9784894146719136782011032591772330616588698650629 4
3183141236334841851779088120360233282957052325831 7
1794949717135562421748095886304059101561741816275 0
6372430552440509130586107235520485592847579364235 7
6024628034664212377839601949771029506076828600446 0
9230861730073523130319643352345732610347023685817 8
2841443182830031237366080354236839216889742383761
8082177034765392656093836807671696102263353151287 2
8850418398744182010857362109000196186093896160246 0
2088568776363979501033526558876797002408567338277 0
4944534279588887636004528374067296959992161212008 8
1908843324216508529595491070765557857624369203455 1

省略

46076619939139036787876060299301996426764725427403
64011297421238171255442319681637731281152043413431
85060894391599524452878044334616995055050095307491
88403737341503747816446538950384710455426239295288
28998531209709108913722340683342671325423513366061
63474488522182030819462026736771976453958939935828
32050459497790182712782471389976977879426526570877
30339646201949991550918186788970663139748645058559
72718469140463791494759075349756092667385993283854
20778335173462088786041537266587943086744729710814
62579154103447702796046390902553975412328182470276
13052411058118309311147944260478608

steps and then halts!

有限だが計算できない数の発見について
Busy Beaver 問題

最近になって、「忙しいビーバー」 Busy Beaver 問題で、とても重要な発見があった。

何がわかったかを述べる前に、少し、状況を整理しよう。

Busy Beaver 問題というのは、N個の状態を保つTuringマシンが、実際に計算可能な最大ステップ数 BB(N) を求めよという問題だ。

まず、N個の状態を持つTuringマシンが何個あるか考えよう。これは、昨日の遷移表を眺めていれば、簡単にわかる。

'B1R' や'H1L'といった「命令」の数は、一文字目が N+1個（状態の数＋停止状態）、二文字目が2個（ 1 か 0 だから）、三文字目が2個（RかLだから）の可能性があるから、全部で、(N+1) x 2 x 2 通りの組み合わせがある。

Turingマシンは、これらのN x 2 個の命令 (一つの状態について、1の場合と 0の場合について、実行する命令を置いたものだから)から構成される。

だから、N個の状態を持つTuringマシンの総数は、2N個のマス目を、4(N+1)個のもので埋めればいいので、4(N+1) の2N乗だということがわかる。

これは、大きな数になるかもしれないが、「有限」な数だ。だから、これをかたっぱしから試していけば、「原理的」には、BB(N)は求まるはず。

と思うかもしれないが、この手続きはうまくいかない。だって、試すべきマシンの数が「有限」でも、「無限」ループにおちいるマシンにぶつかれば、この手続きは「有限」の時間では終わらないから。

ただ、こうしたループの「落とし穴」ではない、もう一つの問題がある。それは、状態の数Nやチェックすべきチューリングマシンの数と比べて、BB(N)が、きわめて急速に巨大な数に膨れ上がるということだ。（単純なループにはまるのと、答えを求めて、えんえんと計算を続けるのは、多分、別のことだ。BB(N) は、そうした最長の数を求める。）

先に、BB(6)が、36,534桁の数字より大きいという話をした。2014年に、Wythagoras（怪しい名前だ）は、BB(7)について、次のような評価を示した。右側の数は、とても巨大な数だが、「有限」な数字である。

$$\mathrm{BB}(7) > 10^{10^{10^{10^{10^7}}}}$$

冒頭に述べた「Busy Beaver 問題での重要な発見」の話をしよう。

今年（2016年）の4月に、Adam Yedidia と Scott Aaronson は、BB(7910)が、普通の集合論の枠組みの中では決定できないことを証明した。

"A Relatively Small Turing Machine Whose Behavior Is Independent of Set Theory"
http://arxiv.org/pdf/1605.04343v1.pdf (Scottの次のblogが参考になる。
http://www.scottaaronson.com/blog/?p=2725)

それ以来、BB(N)が計算不能になる、より小さいNの探索が、インターネット上で活発に行われ、この "logic hack" で、つい二週間ほど前に、BB(1919)も計算不能であることが示されたのである！

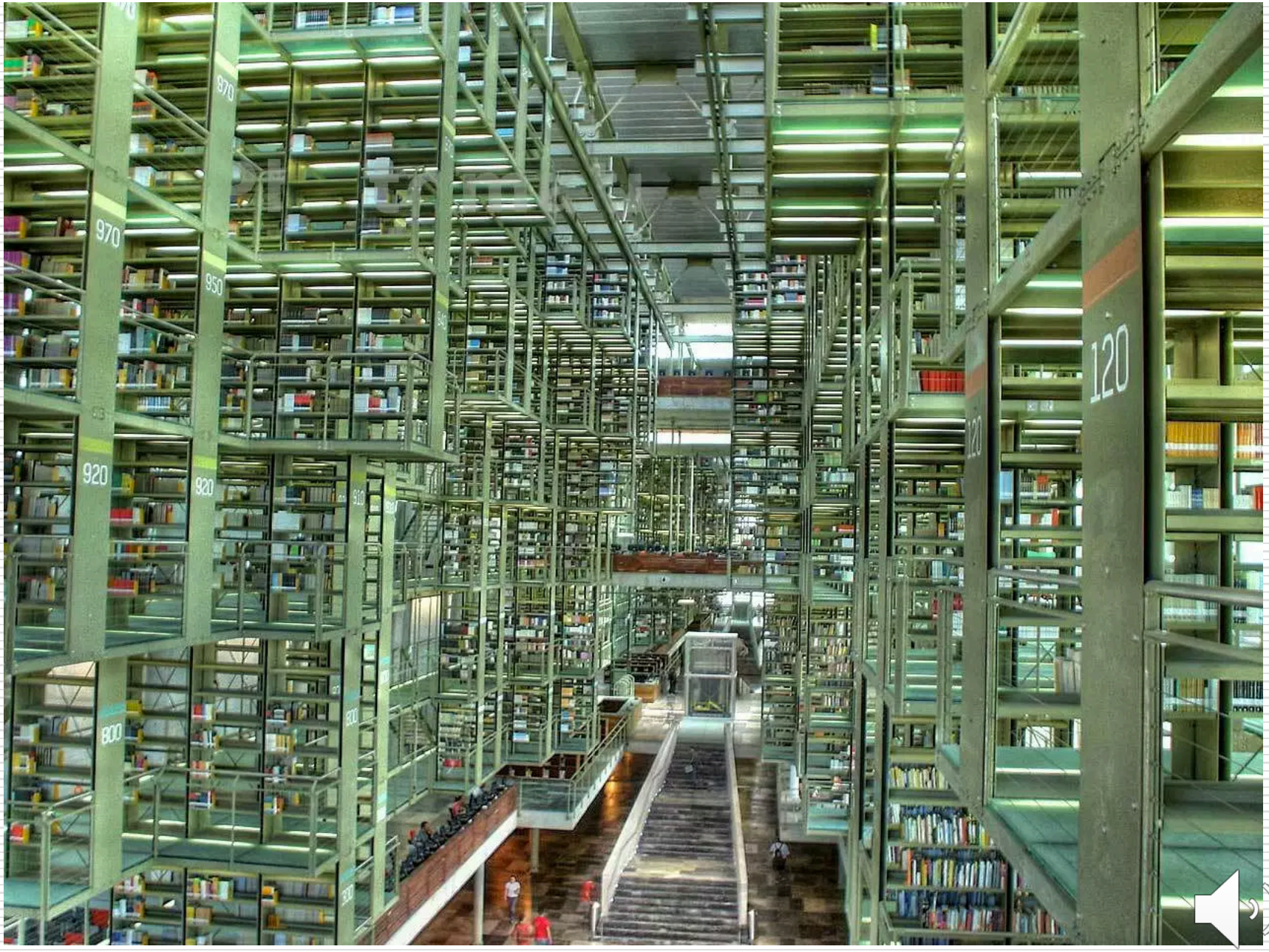
確かに、僕が生きているうちは、BB(10)の値も決まらないだろうと思って、ほぼ、間違いないので、BB(7910) とか BB(1919)とかは、とんでもなく大きな数である。ただ、これらの数字は「有限」で、しかも、BB(N)自体は、とても明確に定義されたものである。

要するに、「有限」の値を持つのだが、どういう数学的な手段を持ちいても、我々が知り得ない数字があるということ。さらに、その限界が、7910とか1919といった、比較的小さな数字で特徴付けられるというのが、驚きである。

計算複雑性理論へ – nと2^n

多項式時間と指数関数的時間

- **Denition. P** is the class of languages L $\subset \{0,1\}^*$ for which there exists a Turing machine M and a polynomial q so that for inputs x $\in \{0,1\}^n$, M terminate in at most q(n) steps and accepts if and only if x $\in$ L.
- **Denition. PSPACE** is dened like P, except we're limited by q(n) space, rather than time.
- **Denition. EXP** is dened like P, but we're limited by $2^{q(n)}$ time steps.


970
950
920
800
120


7/3 マルレク「夜学」
https://yagaku-01.peatix.com/