スーパーサイエンスハイスクール

計算可能性：定規とコンパスによる作図からチューリング機械まで

クップル・ドミニク

山梨大学・コンピュータ理工学

山梨大学
UNIVERSITY OF YAMANASHI
地域の中核 世界の人材

プログラミング

- 現在に、いくつかのプログラミング言語がある
- どのプログラミング言語を知っていますか？

A: なでしこ

B: Java / Kotlin

C: C/C++/C#

D: Python

E: Ruby / Perl

❓ コンピュータで計算できない場合があると思いますか？
❓ プログラミングを始めるとき、まず最初に入力するプログラムは？

```
Hello world!
```

Hello-World 問題

- Hello-World 問題は `Hello, world!` というメッセージを表示するだけの簡単な問題
- プログラミングの入門例としてよく使われる

◎ 正解

```
#!/usr/bin/python3
print('Hello, world!')
```

✗ 誤り

```
#!/usr/bin/python3
print('something')
```

先生の課題

- 生徒が `Hello, world!` というメッセージだけを表示するコードを書いたかどうかを確認する
- `Hello, world!` 以外の出力となった場合は誤り
- 回答を確認するためのプログラムを作る

← ❓ 課題に対して、どのようなプログラムを作ることができるか？

ケース

簡単

```
#!/usr/bin/python3
print('Hello, world!')
```

どちらのプログラムも実行すると、`Hello, world!` が出力されているので、正しい

難しい

```
#!/usr/bin/python3
print("".join(map(lambda x: chr((lambda p, x:
int(sum(map(lambda i: p[i]*x**(11-i),
range(12))) + 0.001))([__import__("struct").unpack
('>d', bytes([int(s[8*i:8*i+8], 2)
for i in range(8)]))[0] for s in
["10111111010100001110101001100110000011010011000001100100011111",
"00111111101100000001011101000010111000000010101111001001011100 11",
"10111111111110101010011110000100011110000001111111001101001000100",
"01000000001110010011011100001101010001100000010101000011000110 00",
"11000000011011100000100000101001101010011110100000001110111011 01",
"01000000100101110110010101110001011100100000100001010101110111 00",
"11000000101101111111101111000011010111000110111010011111111011 00",
"01000000110011111010110100100101110110110010110010000011100000 11",
"11000000110110011000010110010010100111001100010100110101001111 01",
"01000000110101100101010101100100001100011111011111110011001000 111",
"11000000101111111000000110000110001001100101101000010001101101 10",
"01000000010100011111111111111111111111101001001001010111111110 01"]],
x)),
```

(引用: https://gist.github.com/kierendavies/3809373)

一番確認しづらいプログラムの例

```python
#!/usr/bin/env python3

def is_prime(n):
    if(n <= 1):
        return False
    else:
        for i in range(2,n//2+1):
            if(n % i == 0):
                return False
    return True

def is_sum_of_two_primes(n):
    for i in range(2,n//2+1):
        if is_prime(i) and is_prime(n-i):
            return True
    return False

x = 4
while is_sum_of_two_primes(x):
    x = x + 2
```

❓ このプログラムを実行したら、どうなるか？

- 出力されるかどうかわからない
- なぜわからないのか？

予想 (ゴールドバッハ予想). すべての4以上の偶数は2つの素数の和として表すことができる。 すなわち、$\forall x \geq 2 \exists p, q$ 素数 : $2x = p + q$。 （予想は証明されていない主張)

例.

- 4 = 2 + 2 ❓ 次は6=？
- 6 = 3 + 3
- 8 = 5 + 3
- 10 = 7 + 3 = 5 + 5
- 12 = 7 + 5
- 14 = 11 + 3 = 7 + 7
- 16 = 13 + 3 = 11 + 5
- 18 = 13 + 5 = 11 + 7

- ゴールドバッハ予想（1742年）は未解決！
- 左のプログラムが `Hello, world!` を出力する場合、ゴールドバッハ予想は間違っているという証拠になる
- Hello-World 問題の単純さにもかかわらず、それを正しく評価するプログラムはない！（これを証明します）
- 計算可能性理論：理想的なコンピューターによって計算できることとできないことについての理論

コンピュータの計算可能性

- 計算とは何か？計算モデルとは何か？
- 計算モデル C が与えられたとき:
  - C は何を計算できるか？
  - C が計算できないものは何か？
- 目的: 計算を効率的に行う方法は？

❓ 以下の質問を解析する：

- `Hello, World!` 問題を解決できるか？
- プログラムが停止するかどうかを判定できるか？
- アンチウイルスソフトウェアはすべてのウイルスを見つけることができるか？

計算方法と証明システム

計算方法と論理を行うツール

- 計算方法
  - 画線法 (正の文字で数える)
  - ローマ数字
  - 算盤
- コンパスと定規の構成
  - タレスの定理
  - ピタゴラスの定理
  - 黄金比
  - 六角形
- 計算システム
  - アリストテレスのOrganon [紀元前350年]
  - ユークリッドのElements [紀元前300年]
  - チューリング機械 [1936年]

画線法 (正の文字で数える)

1. 一
2. 丅
3. 下
4. 丌
5. 正

- 正の整数を表す
- 加算 (足し算,+)、減算 (引き算, −)、乗算 (掛け算, ×)、除算 (割り算,÷) を行う
- 問題: 大きな数は表しにくい

❓ 正 + 正 + 正 + 正 + 下 - 下 は何ですか？

A: 正 正 正 正 正

B: 正 正 下

C: 正 正 正

D: 正 正 正 正

ローマ数字

- 大きな数を記号で略す
- I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, そして M = 1000
- より大きな値の記号の後に記号が続く場合、値は追加される
- より小さな値の記号の後に記号が続く場合、値は減算される
- 例: IV = 4, IX = 9, XL = 40, XC = 90, CD = 400, そして CM = 900
- 問題: 計算は難しい

❓ XXV + LXXV は何ですか？

A: XL

B: XC

C: L

D: XXX

E: C

❓ MCMXCIX - LXXV は何ですか？

A: MCMXXIV

B: MCMXXV

C: MCMXXVI

D: MCMXXIV

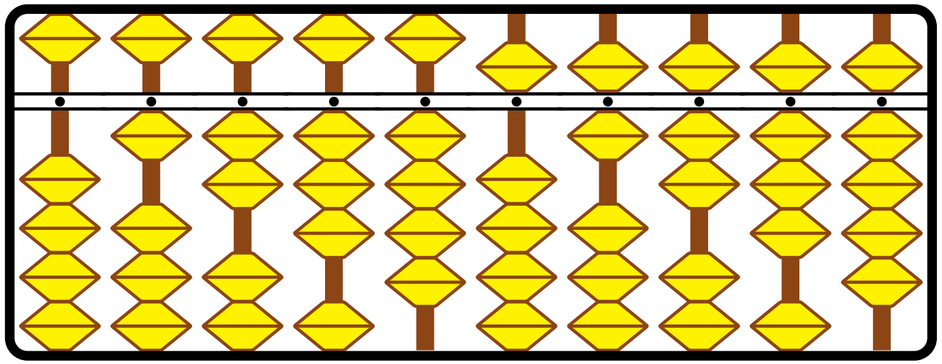
E: MCMXV

- XXV = 25, LXXV = 75, 100 = C
- MCMXCIX = 1999, LXXV = 75, 1924 = MCMXXIV

算盤 (そろばん)

- 下デッキに4つの玉と上デッキに1つの玉
- 各列は0から9の数字xを表す
- 上デッキは$x < 5$を示すフラグ
- 下デッキは残りを一進法でカウント

- 0: $0 \cdot 0$
- 1: $0 \cdot 1$
- 2: $0 \cdot 11$
- 3: $0 \cdot 111$
- 4: $0 \cdot 1111$
- 5: $1 \cdot 0$
- 6: $1 \cdot 1$
- 7: $1 \cdot 11$
- 8: $1 \cdot 111$
- 9: $1 \cdot 1111$



- モダンコンピューター: 上デッキのみ使用 (2進数表記)
- そろばんでどれだけ速くなれるか？

そろばん

DISTRIBUTION: FREE TUESDAY, November 12, 1946

THE WINNAH Kiyoshi "The Hands" Matsuzaki waves his answer sheet triumphantly aloft at the close of the last heat of the Abacus vs. the Calculating Machine Contest, sponsored by the Stars and Stripes at the Ernie Pyle Theater. The abacus victory was decisive..

(Staff Photo by PFC GERALD LICHT)

Abacus Expert Downs Calculating Machine Surrendering Only In Multiplication Race

By PVT. LAWRENCE STERN, Staff Writer

画像出典: www.guidigo.com

そろばんが電卓を打ち負かす

- 1946年11月12日
- 松崎清 (そろばん) 対 トーマス・ネイサン・ウッド (米軍)
- 速さの競争でそろばんオペレーターが勝利

ユークリッド幾何学: コンパスと定規

- 計算から幾何学へ
- 計算モデル C: コンパスと定規
- ❓ C では平面上にどのような図形を描けますか？

基本的な構成要素:

- 頂点を平面上に設定
- ２つの頂点を通る直線を描く
- 頂点を中心に半径が２つの頂点の間の距離の円を描く
- 2直線や円との交差頂点に頂点を描く

ユークリッドの幾何学における数学

数学的システム：

- 公理: 一連の主張
- 導出規則: 公理から新しい主張を導出するための有限の規則
- 主張: 仮定、真偽になる
- 主張の証明:
  - 公理から主張を導出する手順の列
  - 各手順は、公理であるか、前の手順から導出されたものである

システムの特性：

- 整合性: 公理から矛盾が導出されない（すべて主張は両方、真と偽であることを導出されない）
- 完全性: 完全なシステムは、公理から始めて推論規則に従って真の主張をすべて導出できる
- 理想的なシステム は完全かつ整合的である

ユークリッド幾何学

ほとんどの証明は構成による

- 問題: 入力と望ましい出力の記述
- 手続き: 一連のアクションの指定
- アルゴリズム: 必ず終了する手続き

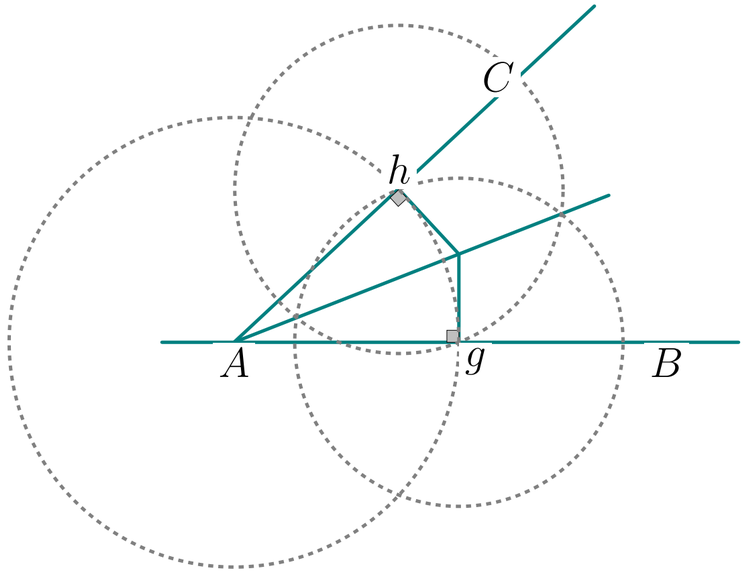
例.

頂点 A で2つの直線 B と C が交差しているとき、頂点 A の角度 α が与えられたとき、$\frac{\alpha}{2}$ を計算します。

証明.

- 頂点 A を中心に円を描く
- 2つの直線と円の交点は2つの頂点 g と h を作成
- 頂点 g と h から同じ半径の円を描くと、交点は2つあり、その直線が角度を二等分します

証明はアルゴリズムです！

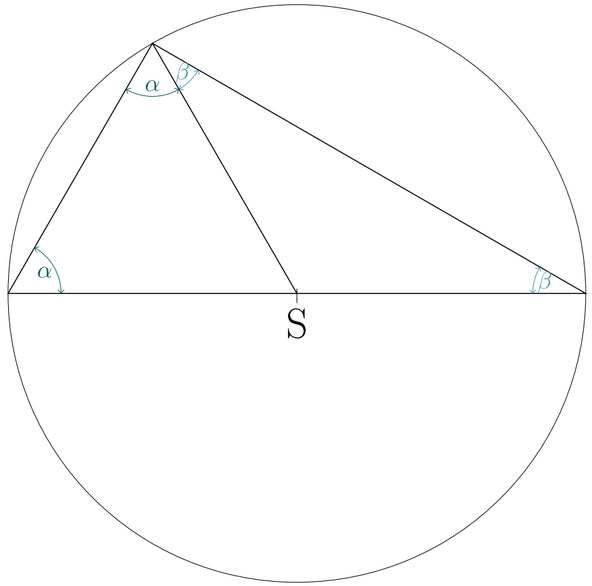


ユークリッド幾何学: 描けるものは何か？

その他の構成:

- タレスの定理
- ピタゴラスの定理
- 黄金比
- 正三角形 (正三角形: すべての辺と角が等しい)
- 正六角形

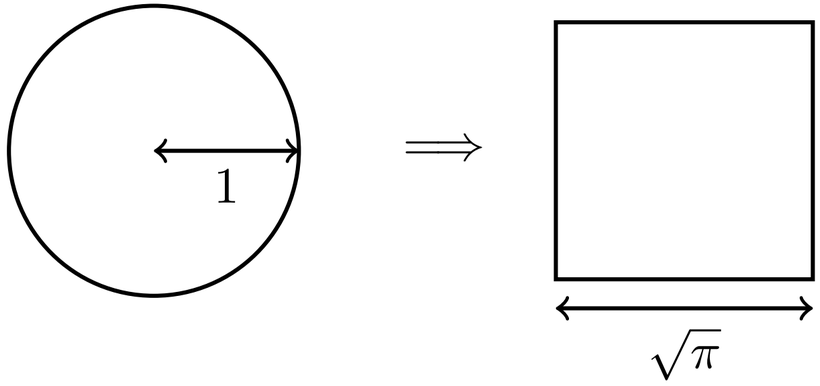
ユークリッド幾何学は整合的ですが、完全でしょうか？



ユークリッド幾何学: 不完全性

リンデマン・ヴェイエルシュトラスの定理 (1882)

- π は 超越数 であり、有理係数の有限多項式の根ではない
- つまり、コンパスと定規で π を計算することはできません
- 問題「円の四角化」は解くことができません:
  - 与えられた円と同じ面積の正方形を構築する
  - 円の半径が1の場合、解は辺が $\sqrt{\pi}$ の正方形が必要です



ヒルベルトのプログラム (1900)

目標: 数学のすべての基礎を提供する

- 完全性: すべての真の数学的な主張が形式化できることの証明
- 正当性: 形式化で矛盾が生じないことの証明
- 計算可能性: 任意の数学的な主張の真偽を判定するアルゴリズムが存在すること

カントール: 無限集合論 (1874)

2つの集合が同じ濃度を持つとは、それらを全単射で配置できる場合

$\mathbb{N}$ と $\mathbb{E}$ は同じ濃度を持つか？

- $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, \ldots\}$, 自然数
- $\mathbb{E} = \{0, 2, 4, 6, 8, 10, 12, \ldots\}$, 偶数の自然数
- どうすれば $\mathbb{E}$ と $\mathbb{N}$ が同じ濃度になるでしょうか？
- 目標: $\mathbb{E}$ と $\mathbb{N}$ の間の全単射関数 $f : \mathbb{E} \to \mathbb{N}$ を作成する
- 例: $f(x) := x$ は失敗します! ($f(0) = 0$, $f(2) = 2$, $f(4) = 4$, ...)
- しかし: $f(x) := \frac{x}{2}$ は全単射です! ($f(0) = 0$, $f(2) = 1$, $f(4) = 2$, ...)

ノート

- カントールの定義は、単一の全単射だけを要求しており、すべての全単射が全単射である必要はありません！
- 集合が有限の場合、この違いは決して生じません

$\mathbb{N}$ と $\mathbb{Z}$ は同じ濃度を持つか？

- $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, \ldots\}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$

❓ $\mathbb{N}$ と $\mathbb{Z}$ は同じ濃度を持つか？

目標: $\mathbb{N}$ から $\mathbb{Z}$ への全単射関数 $f : \mathbb{N} \to \mathbb{Z}$ を作成する

はい:

$$f(x) = \begin{cases} \lceil x/2 \rceil & \text{ただし } x \text{ は奇数} \\ -\lceil x/2 \rceil & \text{ただし } x \text{ は偶数} \end{cases}$$

- $f(0) = 0,$
- $f(1) = 1,$
- $f(2) = -1,$
- $f(3) = 2,$
- $f(4) = -2,$
- $f(5) = 3,$
- $f(6) = -3, \ldots$
- $\mathbb{Z} = \{0, 1, -1, 2, -2, 3, -3, \ldots\}$

推移性

補題. $A \to B$ および $B \to C$ の両方が全単射である場合、$A \to C$ は全単射である

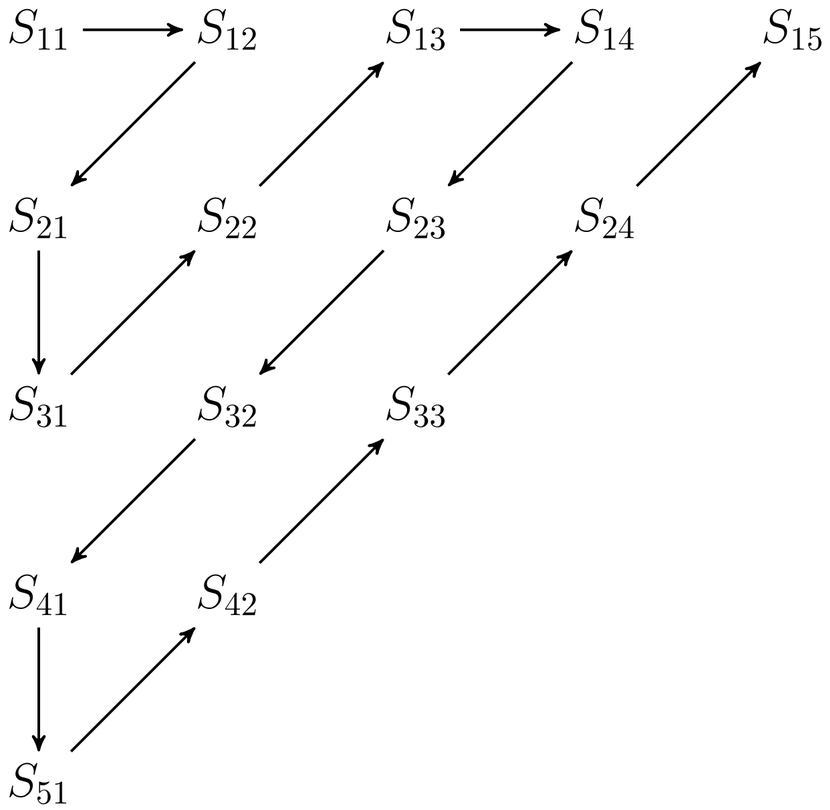
したがって、$\mathbb{N}$、$\mathbb{E}$、および$\mathbb{Z}$は同じ濃度を持ちます

定義 (可算集合). 集合 X が可算とは、自然数 $\mathbb{N}$ から X までの全単射関数がある

例. $\mathbb{N}$、$\mathbb{E}$、および$\mathbb{Z}$は可算である

$\mathbb{N}$ と $\mathbb{Q}$ は同じ濃度を持つか？

- $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, \ldots\}$
- $\mathbb{Q} = \left\{ \frac{p}{q} \mid p, q \in \mathbb{N}, q \neq 0 \right\}$, 有理数
- $x = \frac{p}{q} \in \mathbb{Q}$ を選択
- その後、右の曲線を通じてノード S_{pq} を見つける
- (全単射にするためにいくつかの頂点を省く必要です。)



$\mathbb{N}$ と $\mathbb{R}$ は同じ濃度を持つか？

- $\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, \ldots\}$
- $\mathbb{R}$ = 実数、つまり、10進展開で表現できるすべての数

例.

- $0.33333\ldots = 0.\bar{3} = \frac{1}{3} \in \mathbb{R} \cap \mathbb{Q}$

$\mathbb{R}$ には $\mathbb{Q}$ に含まれない数が持つ:

- 円周率 $\pi = 3.14159\ldots$
- シャンパーノウン数:
  - $\sum_{n=1}^{\infty} 10^{-n(n+1)/2} = 0.1010010001000010000010000001\ldots$
  - i-番目の桁と$(i+1)$-番目の間にi個のゼロがあります
- リウビルの定数:
  - $0.11000100000000000000000001\ldots$
  - $(i!)$番目の桁はiです

非有理数は無限で繰り返さない10進展開を持つ

❓ $\mathbb{R}$ は可算？

カントールの対角線法

定理. 閉区間 $[0,1] \subset \mathbb{R}$ は可算ではない

証明 (背理法).

- 仮定: $[0,1] \subset \mathbb{R}$ は可算
- f を $\mathbb{N}$ から $[0,1]$ への全単射とする
- $L[i][j]$ は $f(i)$ の j 番目の桁を格納する
- したがって、$L[0]$ は $f(0)$ の桁、$L[1]$ は $f(1)$ の桁、などを格納する
- x を次のように定義する:
  - $x = 0.d_0 d_1 d_2 d_3 d_4 d_5 \ldots$
  - ここで、$d_i = 1$ ならば $L[i][i] \neq 1$ であり、$d_i = 0$ ならば $L[i][i] = 1$
- 各 i に対して、x は $f(i)$ と$i 番目の桁で異なる
- したがって、x はリスト $f(0), f(1), f(2), \ldots$ にない
- 矛盾: f は全単射ではない

例.

- $f(0) = 0.2076\cdots$
- $f(1) = 0.4389\cdots$
- $f(2) = 0.3650\cdots$
- $f(3) = 0.6270\cdots$

L	小数点以下の位置				
index	0	1	2	3	$\cdots$
0	2	0	7	6	$\cdots$
1	4	3	8	9	$\cdots$
2	3	6	5	0	$\cdots$
3	6	2	7	2	$\cdots$
$\vdots$					$\ddots$

L	小数点以下の位置				
index	0	1	2	3	$\cdots$
0	d_0				$\cdots$
1		d_1			$\cdots$
2			d_2		$\cdots$
3				d_3	$\cdots$
$\vdots$					$\ddots$

ラッセルのパラドックス

- カントールの集合論の問題
- S を次のように定義する:

$$S := \{\text{集合 } R \mid R \notin R\}$$

❓ S は自分自身の要素ですか？

- S が自分自身の要素である場合、S の定義により、S は自分自身の要素ではない
- S が自分自身の要素でない場合、S の定義により、S は自分自身の要素

結果

理論は不完全!

より簡単なパラドックス

山梨大学の教師として、次の主張 S を述べます:

> 山梨大学の教師は嘘をつく!

❓ S は真か偽か？

- S が真である場合、S によると私は嘘つきであり、S が真であると述べた
- S が偽である場合、S によると私は嘘をついていないので、S は真

(出典: クレタの哲学者エピメニデス、紀元前600年頃)

最も簡単なパラドックス

主張 S:

> S は偽

❓ S は真か偽か？

- S が真である場合、S は偽であると述べた
- S が偽である場合、S は真であると述べた

ゲーデルは、ヒルベルトのプログラムにそのようなSが含まれていることを示した！

ゲーデルの不完全性定理 (1931)

ヒルベルトのプログラムの目標は不可能であることを示す

定理. 理想的なシステムが存在ない

アイデア:

- 主張 G を次のように定義する: G はシステムでの証明を持たない
- ラッセルのパラドックスのように、G は矛盾につながる

1. G が証明される場合、

- G はシステムでの証明を持つ
- したがって、システムは矛盾している

2. G が証明されない場合、

- G は真の主張
- したがって、システムは不完全です

ゲーデル: 素朴ではないシステムはこのような主張 G を持つ

計算不可能性

- 問題が 計算可能 である場合、その問題を解くアルゴリズムが存在する (ほぼ無限のRAM、CPU、GPUなどがあると仮定)
- それ以外の場合、計算不可能 と呼ぶ
- チューリングは計算不可能な問題が存在することを証明した

問題 (停止問題).

- 入力: 手続きを表す文字列
- 出力: 評価が終了するかどうか `true` または `false`

❓ 停止問題を解く関数 h を作成できますか？

停止問題

- h を探す：
  - 入力: 手続き P
  - 出力する： P が停止すれば `true`、そうでなければ `false` を返す

例.

- $h(3+5) =$ `true`
- $h($ 永遠にループする $) =$ `false`
- $h($ wait(∞) $)$ = `false`

`paradox` 関数の定義

```
paradox(P) {
if h(P) then
  loop forever; // 停止しない
else
  return true; // 停止する
}
```

❓ `paradox(paradox)` は停止しますか？

1. `paradox(paradox)` が停止すると仮定する:
   - この場合、$h($`paradox`$) =$ `true`
   - `paradox` は入力 `paradox` で永遠にループする (停止しない)
2. `paradox(paradox)` が停止しないと仮定する:
   - この場合、$h($`paradox`$) =$ `false`
   - `paradox` は入力 `paradox` で停止する

結論

- `paradox`は矛盾に導く
- hが矛盾に導くことを示していない（しかし強く示唆している）
- 証明には普遍チューリング機械が必要

チューリングの計算モデル

目的:

- 人間の計算のモデル
- 不要な詳細を割愛する

ツール:

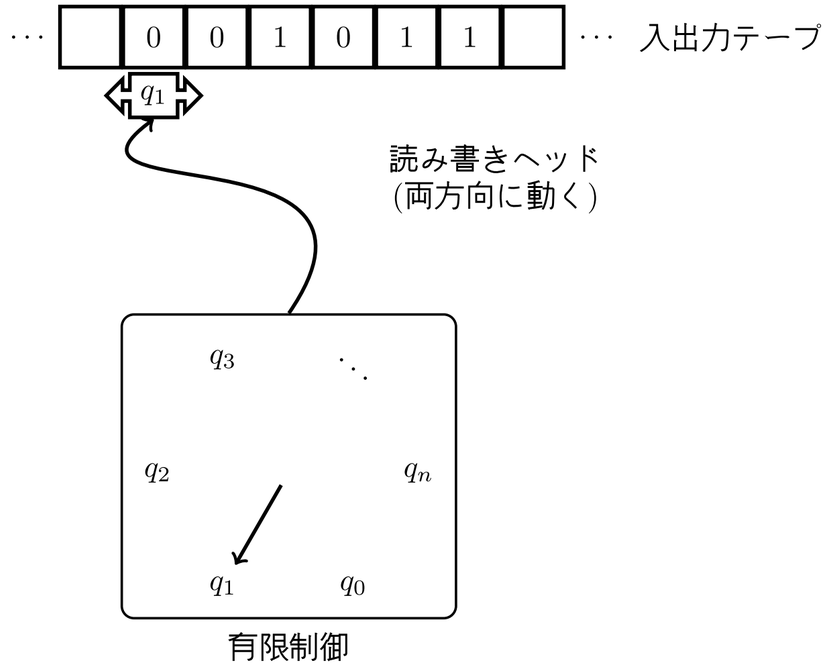
- 有限制御 (脳)
- 長期記憶 (紙またはテープ)

操作:

- 紙に書き込みまたは読み取り
- 紙の別の部分に注意を移す

アクションのトリガー:

- 限られた数の状態 $q_0, q_1, \ldots, q_n$
- 次に何をするかは、紙のどの部分にいるかとアクティブな状態に依存する



何が計算できるか？

- 計算 (加算、乗算など)
- 回分の確認
- π の最初の桁の2進表現の計算

実際の詳細は重要ではない:

- 複数のテープ
- 両方向に無限のテープ
- アルファベット / 数字システムのサイズ (紙に書かれるもの) — 有限である限り

コンウェイのライフゲーム

セルオートマトン

- セルのグリッド; 各セルには生または死の2つの状態がある
- ラウンド中、すべてのセルが現在の状態に基づいて再計算される

ルール

1. 過疎 (かそ): 2つ未満の生きた隣のセルを持つ生きたセルは死ぬ
2. 継続 (けいぞく): 2つまたは3つの生きた隣のセルを持つ生きたセルは生き続ける
3. 過密 (かみつ): 3つ以上の生きた隣のセルを持つ生きたセルは死ぬ
4. 繁殖 (はんしょく): 死んだセルがちょうど3つの生きた隣のセルを持つと生きる

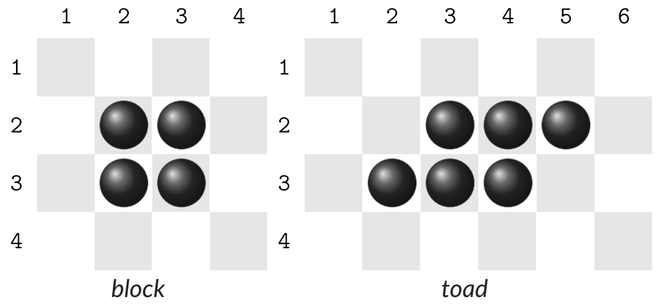
コンウェイのライフゲーム

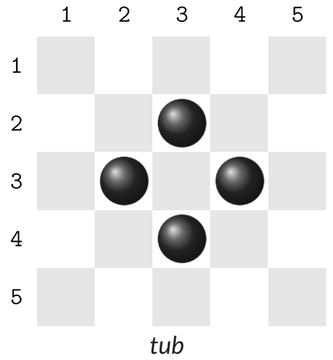
ルール

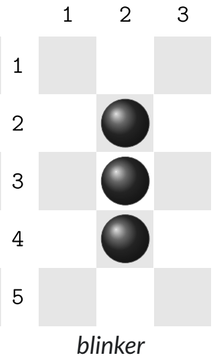
1. 過疎 (かそ): 2つ未満の生きた隣のセルを持つ生きたセルは死ぬ
2. 継続 (けいぞく): 2つまたは3つの生きた隣のセルを持つ生きたセルは生き続ける
3. 過密 (かみつ): 3つ以上の生きた隣のセルを持つ生きたセルは死ぬ
4. 繁殖 (はんしょく): 死んだセルがちょうど3つの生きた隣のセルを持つと生きる

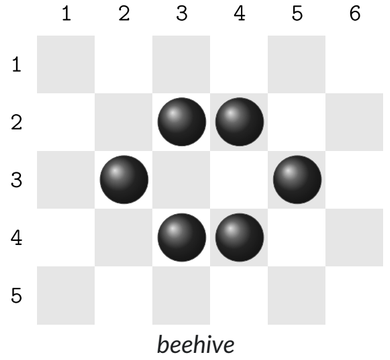
課題

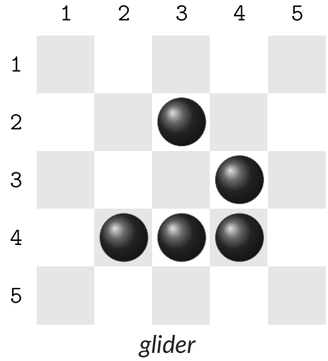
- 右のパターンの最初の4ラウンドを計算しましょう
- ❓ どれは繰り返しますか？







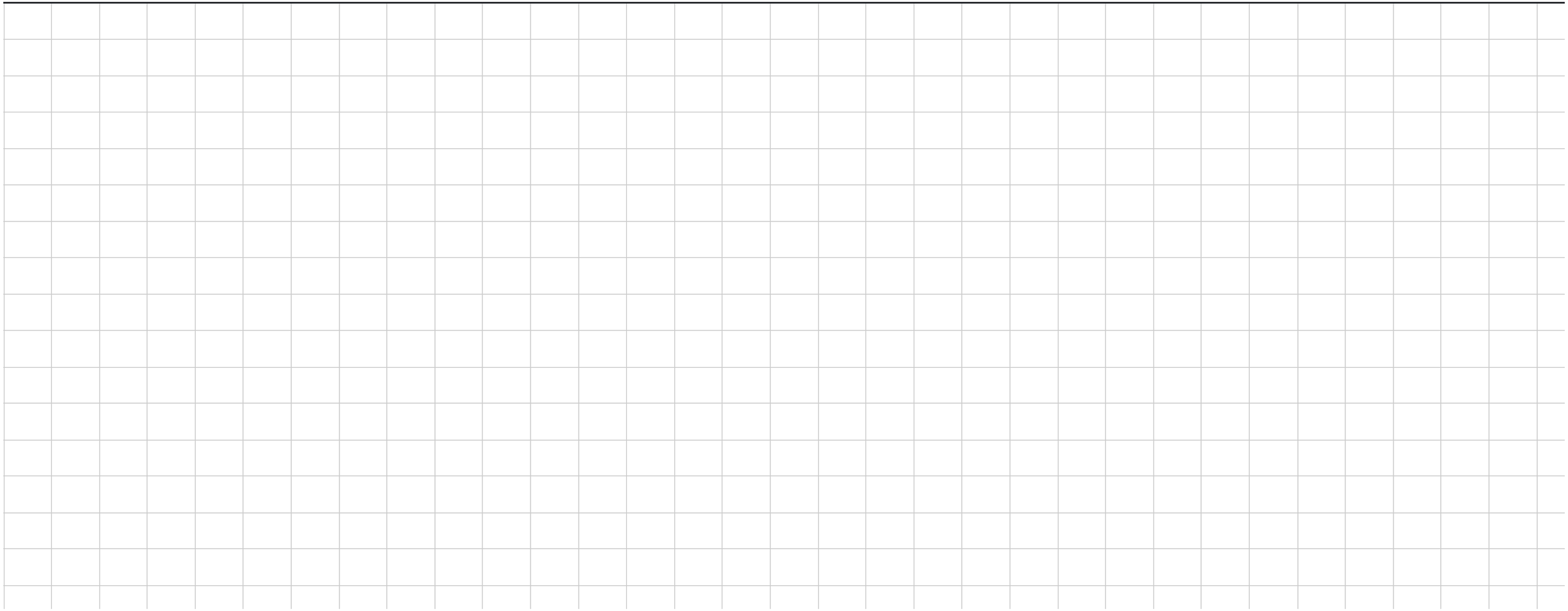




コンウェイのライフゲーム

1. 過疎 (かそ): 2つ未満の生きた隣のセルを持つ生きたセルは死ぬ
2. 継続 (けいぞく): 2つまたは3つの生きた隣のセルを持つ生きたセルは生き続ける
3. 過密 (かみつ): 3つ以上の生きた隣のセルを持つ生きたセルは死ぬ
4. 繁殖 (はんしょく): 死んだセルがちょうど3つの生きた隣のセルを持つと生きる

start reset



コンウェイのライフゲーム

- チューリング機械でシミュレートできる!

由来: Let's build a computer in Conway's game of life ∴, Alan Zucconi, https://www.youtube.com/watch?v=Kk2M

チャーチ・チューリングのテーゼ

チャーチ・チューリングのテーゼ: アルゴリズムで計算できる関数はすべてチューリング機械で計算できる

- テーゼは証明または反証できる主張ではない
- 人間の脳と鉛筆/紙で把握して実行できるアルゴリズムは、メモリに制限がない通常のデジタルコンピューターで実行できる
- 誰もチャーチ・チューリングのテーゼに反する例を提供してない — 一貫した方法で人間が計算するが、コンピューターでプログラムできない具体例を提供してない

多くの「合理的で物理的に実現可能な」計算モデルはチューリング機械でシミュレートできる

- 算盤 (そろばん)
- ユークリッド幾何学
- 現代のコンピューター
- 人間の脳?

チューリング機械の列挙

チューリング機械Mは次のように記述できる

- 有限のアルファベットΣ
- 有限の状態
- 有限の遷移ルール
- 有限の初期入力

Σ^*: Σのすべての有限文字列、例えば

- Java/C++/Pythonのような言語で書かれたすべてのコード
- 自然言語テキストなどのすべての有限の初期入力
- Σ^*は可算

したがって、すべてのチューリング機械$M_0, M_1, M_2, \ldots$は可算!

- したがって、Mを(巨大な)数$\langle M \rangle \in \mathbb{N}$として記述できる
- したがって、すべての可能なチューリング機械を列挙できる
- (すべての数が有効なチューリング機械に対応するわけではない)

カントールの対角線法の証明:

- チューリング機械で生成できない実数$\in \mathbb{R}$が存在する

普遍チューリング機械

普遍チューリング機械 U:

- 入力: チューリング機械Mの数$\langle M \rangle$
- Mをシミュレートする
- Uのテープには最初に$\langle M \rangle$が格納されるため、コード/プログラムが入力の一部!

ほとんどのプログラミング言語は普遍チューリング機械として使用できる

停止問題の計算不可能性の証明

数え上げによる矛盾

- テーブル$L[i][j]$を作成して、M_iが$\langle M_j \rangle$の入力で停止するかどうかを格納
- $d_i := h(M_i) \in \{$`true`, `false`$\}$と定義
- `paradox`というチューリング機械を定義する: `paradox`($\langle M_i \rangle$)が停止する $\iff$ d_i = `false`
  - `paradox`はLの対角線を否定する
  - `paradox`はLに含まれません

したがって: チューリング機械で計算不可能

L	すべてのチューリング機械を入力として			
index	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\cdots$
M_0	$h(M_0(\langle M_0 \rangle))$	$h(M_0(\langle M_1 \rangle))$	$h(M_0(\langle M_2 \rangle))$	$\cdots$
M_1	$h(M_1(\langle M_0 \rangle))$	$h(M_1(\langle M_1 \rangle))$	$h(M_1(\langle M_2 \rangle))$	$\cdots$
M_2	$h(M_2(\langle M_0 \rangle))$	$h(M_2(\langle M_1 \rangle))$	$h(M_2(\langle M_2 \rangle))$	$\cdots$
$\vdots$				$\ddots$

L	すべてのチューリング機械を入力として			
index	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\cdots$
M_0	d_0			$\cdots$
M_1		d_1		$\cdots$
M_2			d_2	$\cdots$
$\vdots$				$\ddots$

オラクルと帰着

オラクルは問題を解決できるブラックボックス

- デルフォイの神託: どんな質問でも答えることができた (紀元前1400年)
- 停止問題を計算できる!
- しかし、存在は不明

問題の帰着にオラクルを使用:

- 帰着: 問題 Pの解を使用して異なる問題 Qを解決できることを示す
- 問題 Qが問題 Pに帰着可能である場合、Pの解を使用してQを解決できる
- PからQへの帰着は、Pを計算することがQを計算することと少なくとも同じくらい困難であることを示す

入力のないプログラム

- K: すべての入力で停止するチューリング機械の集合
- K_0: 入力を持たない停止するチューリング機械の集合
- OでK_0のオラクルが与えられる
- チューリング機械Mが与えられたとき、Oは$M \in K_0$かどうかをテストできる

課題

- Oを使用してKをテストできますか？
- チューリング機械Mと入力Iを持つとき

❓ $M(I) \in K$ですか？

- $\langle M(I) \rangle$はIがハードコードされたMのエンコーディング
- したがって、$\langle M(I) \rangle$は入力なしのチューリング機械を符号化
- したがって、$\langle M(I) \rangle \in K_0 \; \forall I \iff M \in K$
- Kのメンバーシップを判定する問題をK_0のメンバーシップを判定する問題に帰着
- K_0が計算可能であればKも計算可能
- しかし: Kが計算不可能であることはすでに知っているので、K_0も計算不可能

(計算可能: チューリング機械で計算可能)

Hello-World問題は計算可能か？

これからは、チューリング機械が含んでいるプログラムという用語を使用する

- H: `Hello, world!`を出力して停止するプログラムの集合
- O: Hのオラクル

帰着：

- 任意のプログラムPを取る
- すべての出力ステートメントを削除する (副作用がないと仮定)
- Pが最後に`Hello, world!`を出力するようにする
- Oに変更されたプログラムがHに含まれているか尋ねる

これはK_0のオラクルを与えることになります!

```python
#!/usr/bin/env python3

def is_prime(n):
    if(n <= 1):
        return False
    else:
        for i in range(2,n//2+1):
            if(n % i == 0):
                return False
    return True

def is_sum_of_two_primes(n):
    for i in range(2,n//2+1):
        if is_prime(i) and is_prime(n-i):
            return True
    return False

x = 4
while is_sum_of_two_primes(x):
    x = x + 2
```

ウイルス検出

- V: ウイルスのプログラムの集合
  - ウイルス: 他のプログラムを感染させるプログラム
  - 感染したプログラムはウイルスを実行する
- `infect`はプログラムを感染させる関数
- `detect`はプログラムがウイルスであるかどうかを検出する関数
- その場合、`detect(infect) = true`

しかし、hを定義できます:

- $h(\langle M \rangle)$ = detect($\langle M' \rangle$), ここでM'はMの最後に`infect`が追加されたプログラム
- `detect`のオラクルを持つと、停止問題に答えることができる (入力がウイルスでない場合)!
- したがって、`detect`はチューリング機械では計算不可能

まとめ

計算モデル:

- コンパスと定規 (ユークリッド幾何学)
- チューリング機械 (人間と機械の計算をモデル化)
- オラクル (強力すぎる)

チューリング機械による計算不可能性:

- 停止問題
- Hello-World問題
- ウイルス検出