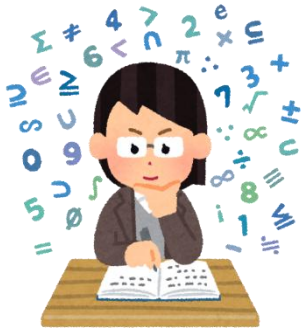
第一日

計算可能性と計算量

令和7年5月19日



計算とは？

けい - さん【計算】

1. 物の数量をはかり数えること。勘定。「――が合う」
2. 加減乗除など、数式に従って処理し数値を引き出すこと。演算。「損失額はざっと――しても一億円」
3. 結果や成り行きをある程度予測し、それを予定の一部に入れて考えること。「多少の失敗は――に入れてある」「――された演技」「――外」

（小学館『デジタル大辞泉』）

本日のテーマ　計算可能性理論 (1930年代～)

計算できるとはどういうことか　数学的に扱う理論

cf.　電子計算機の出現　1940年代～

計算して問題を解くことの難しさを論じたいので……

まず　**問題**とは？

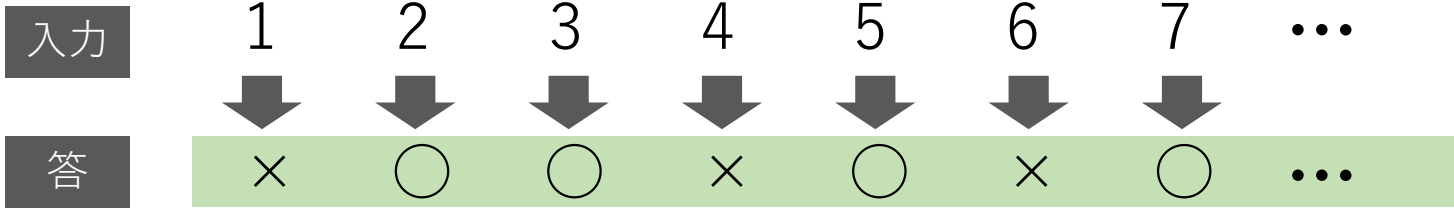
ここでは　各入力に対して　答が○か×か定めたもの
（入力は文字列で表される）
と考えることにする

例えば

問題 PRIME

0 1 … 9 からなる文字列

与えられた正整数（十進法で書く）が素数か答えよ



問題 ABA

a と b からなる

与えられた文字列に「aba」が現れるか答えよ

計算して問題を解くことの難しさを論じたいので……

まず　**問題**とは？

ここでは　各入力に対して　答が○か×か定めたもの
（入力は文字列で表される）
と考えることにする

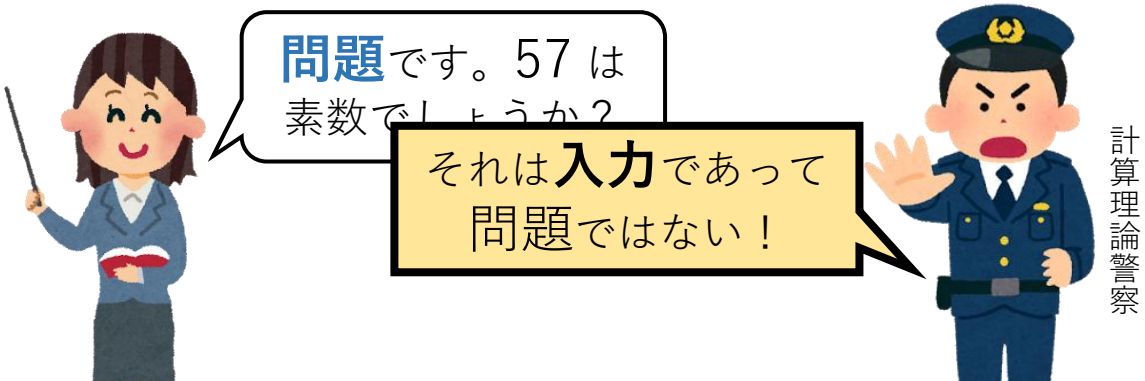
例えば

問題 PRIME　与えられた正整数（十進法で書く）が素数か答えよ

0 1 … 9 からなる文字列

入力	1	2	3	4	5	6	7	…
答	×	○	○	×	○	×	○	…

問題とは
コレ全体のこと！



計算して問題を解くことの難しさを論じたいので……

まず　**問題**とは？

文字列 u の後に v を付けた文字列を uv
文字列 $\underbrace{uu\cdots u}_{m\text{ 個}}$ を u^m などと表すことにする

定義

有限個の文字からなる集合 Σ を考える
Σ に属する文字を有限個並べて得られる文字列の全体を Σ^* と書く
Σ^* の部分集合を**言語**という

この講義では　与えられた文字列が言語に属するか問う問題のみを考える

例　文字集合 $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ を考える（今後は一々断らず適当に定める）

Σ 上の文字列（36 とか 119 とか $\overset{\text{空文字列}}{\varepsilon}$ とか）の全体を Σ^* で表す

文字列のうち十進法で素数を表すもの全体が
言語 PRIME $= \{2,3,5,7,11,13,\ldots\} \subseteq \Sigma^*$ である

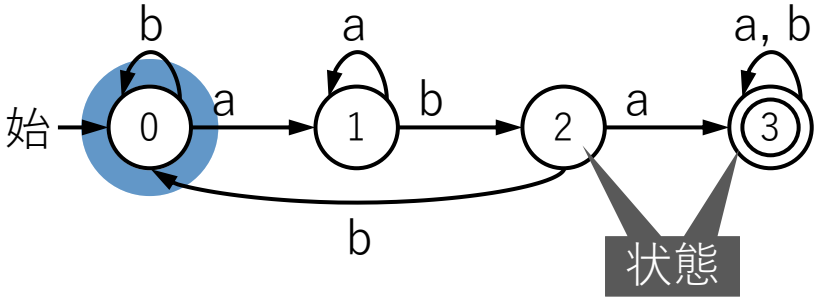
この講義では「文字列を入力すると　それが PRIME に属するか
答えてくれる計算機械」などについて考えたい

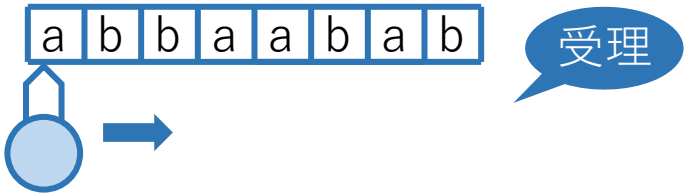
この「入力された文字列が言語 PRIME に属するか判断せよ」という問題
のことも PRIME と呼ぶ（**言語と問題は同じものと考える**）ことにする

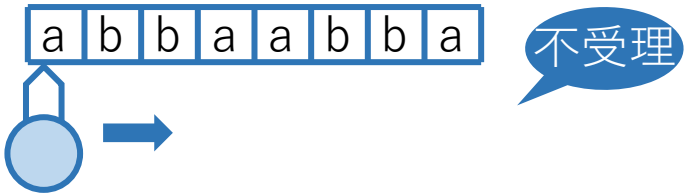
問題
ABA

与えられた文字列に「aba」が現れるか答えよ

この問題を解く有限状態機械







状態は有限個
矢印も有限本

➡ 有限の記述で書き表せる

現在の状態

読んだ文字	0	1	2	③
a	1	1	3	3
b	0	2	0	3

（次にどの状態に行くか記した表）

定義

有限状態機械は次のものにより指定される

- 有限個の**状態**の集合 Q　但し次のものが定まっている
  - 始状態 $q_{始} \in Q$
  - 受理状態の集合 $Q_{受理} \subseteq Q$
- 有限個の**文字**の集合 Σ
- **遷移規則**と呼ばれる函数 $\delta: Q \times \Sigma \to Q$

初め機械は始状態 $q_{始}$ にあり　与えられた文字列 $x \in \Sigma^*$ の左端から
次のこと（遷移）を繰返す
　状態 q で文字 σ を読むと
　状態を $\delta(q, \sigma)$ に変えて一つ右の文字に進む
右端まで終ったとき状態が $Q_{受理}$ に属すれば機械は x を**受理**したという

定義

機械 M が言語 A を**認識**するとは　任意の入力 x に対し

- $x \in A$ のとき M は x を受理し　かつ
- $x \notin A$ のとき M は x を受理しない

すべての入力で
正しく判断！

入力	ε	a	b	aa	ab	ba	bb	aaa	…
問題が要求する 答	○	×	×	○	○	×	○	×	…
機械による計算の結果	受理	不受理	不受理	受理	受理	不受理	受理	不受理	

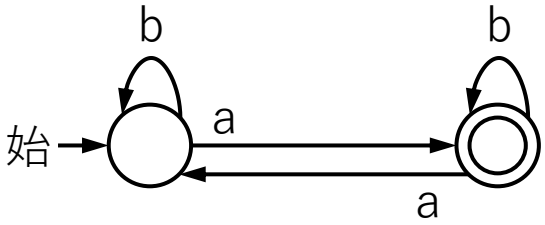
問 1　a と b からなる文字列のうち

（1）a が奇数回現れるもの全体

（2）左から 3 文字目が a であるもの全体

（3）右から 3 文字目が a であるもの全体

をそれぞれ認識する有限状態機械を作って下さい

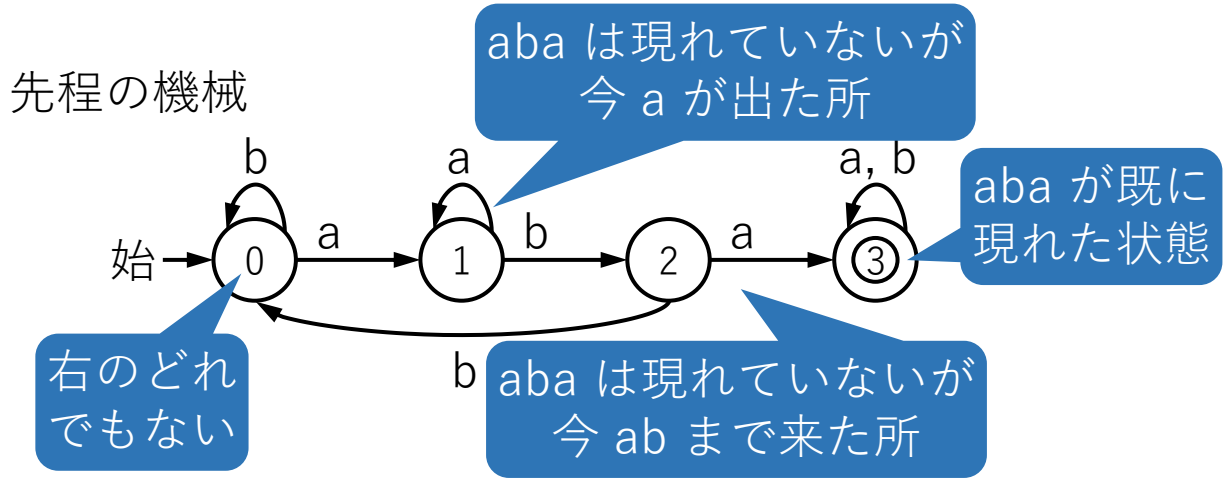


（1）の答（例）

有限状態機械の限界

ABA はなぜ認識できたか？

各時点で「今まで読んだ部分について覚えておくべき情報」が有限種類しかないから



そういう単純な言語しか認識できない　　例えば次の言語は無理

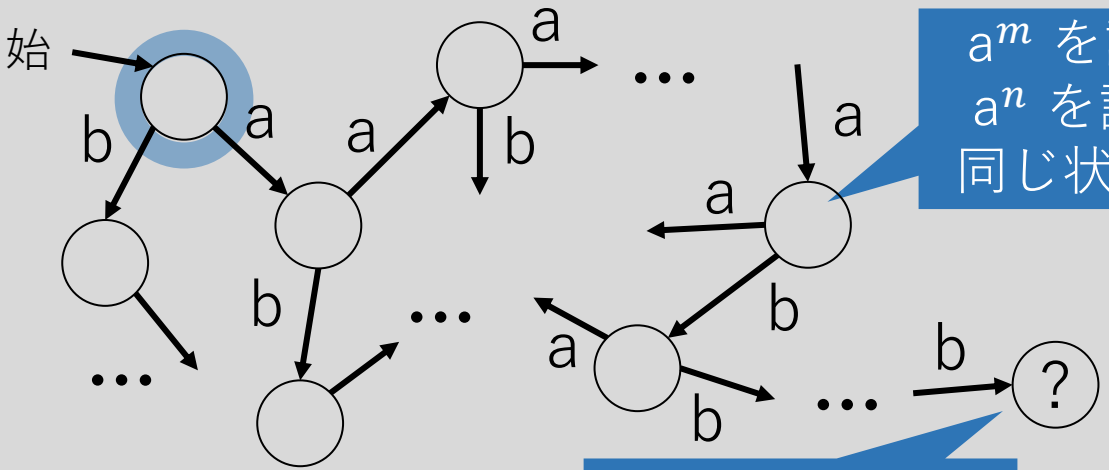
問題
MOREA

与えられた文字列に a が b よりも多く現れるか答えよ

定理

MOREA を認識する有限状態機械は存在しない

証明　そのような機械があるとする
状態は有限個なので　或る数 m と n（$m < n$ とする）が存在して

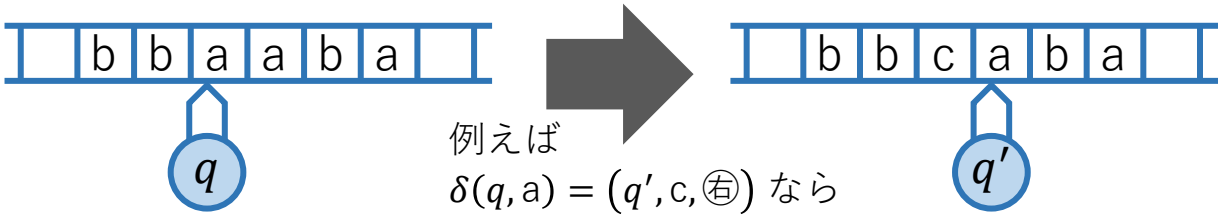


a^m を読んでも
a^n を読んでも
同じ状態になる

$a^m b^m$ は不受理
$a^n b^m$ は受理

では　その後 b^m を読んで至る状態　は◎か？　➡矛盾

読むばかりでなく
書く機能（と
左右に動く機能）
があれば？



定義

チューリング機械は次のものにより指定される

- 有限個の**状態**の集合 Q　但し次のものが定まっている
  - 始状態 $q_{始} \in Q$
  - 受理状態の集合 $Q_{受理} \subseteq Q$
- 有限個の**文字**の集合 Σ　これと空白文字 ␣ を含む集合 $\Gamma \supseteq \Sigma \cup \{␣\}$
- **遷移規則** $\delta: Q \times \Gamma \to (Q \times \Gamma \times \{㊧, ㊨\}) \cup \{\textbf{止}\}$

初め機械は始状態 $q_{始}$ にあり　テープ上に与えられた文字列 $x \in \Sigma^*$ の左端から始めて　次のこと（遷移）を繰返す

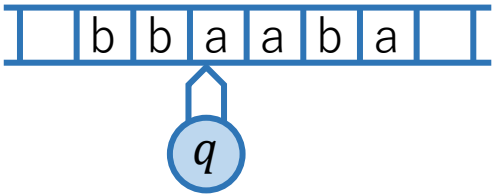
　状態 $q \in Q$ で文字 σ を読むと　$\delta(q, \sigma)$ が

- **止** ならば停止する
- (q', σ', d) なら　状態を q' にし σ' を書込み d の向きに一歩進む

$Q_{受理}$ に属する状態で停止したら機械は x を**受理**したという

これを厳密に定義すると……

※停止せず永遠に動き続ける場合は　受理していないと考える

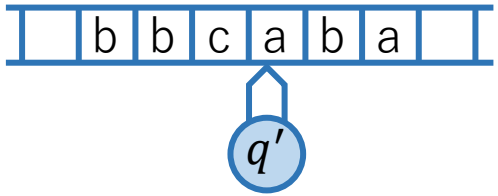


この状況を　bb$q$aaba
のように表すことにする

M

状況の遷移

例えば $\delta(q, \mathrm{a}) = (q', \mathrm{c}, ㊨)$ なら…



次の状況　bbcq'aba

Γ^* の文字列の途中に Q の元がちょうど 1 回だけ現れる文字列を**状況**と呼び（但し先頭や末尾に ␣ を加えた文字列も同じ状況とみなす）状況の遷移 $\xrightarrow{M}$ を次で定義する
状態 $q \in Q$ と文字 $\sigma \in \Gamma$ に対し

- もし $\delta(q, \sigma) = (q', \sigma', ㊧)$ ならば　任意の $u, v \in \Gamma^*$ と $\tau \in \Gamma$ に対し $u\tau q\sigma v \xrightarrow{M} uq'\tau\sigma' v$
- もし $\delta(q, \sigma) = (q', \sigma', ㊨)$ ならば　任意の $u, v \in \Gamma^*$ に対し $uq\sigma v \xrightarrow{M} u\sigma' q' v$

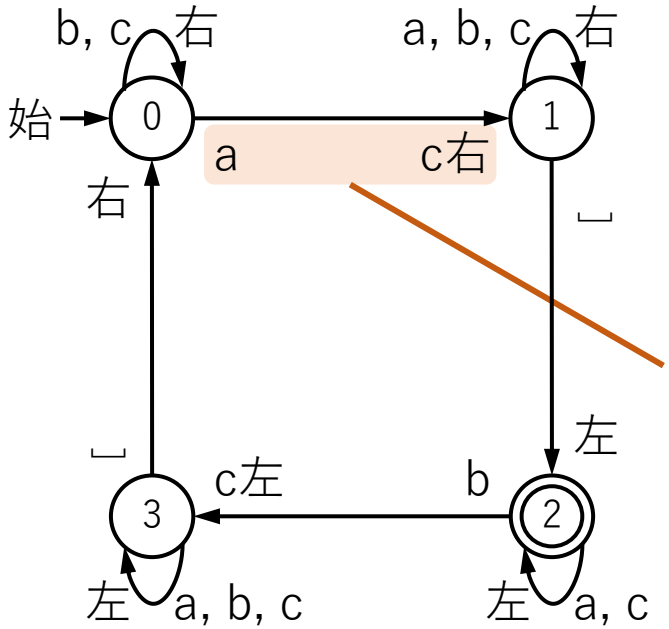
文字列 $x \in \Sigma^*$ を機械 M が**受理**するとは
状況の有限列 $(s_0, \ldots, s_f)$ であって次を満すものが存在することをいう

- $s_0 = q_{始} x$
- $s_0 \xrightarrow{M} s_1 \xrightarrow{M} s_2 \xrightarrow{M} \cdots \xrightarrow{M} s_f$
- s_f には $\delta(q, \sigma) =$ **止** かつ $q \in Q_{受理}$ なる $q\sigma$ が現れる

問題
MOREA

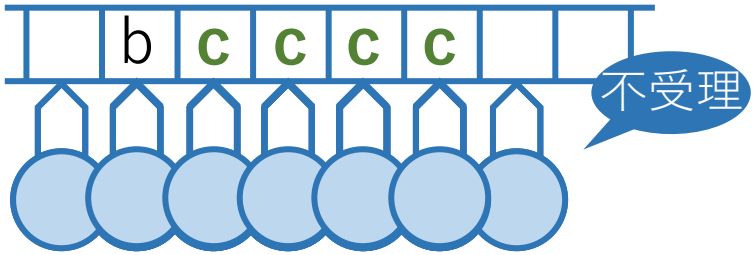
与えられた文字列に
a が b よりも多く現れるか答えよ

MOREA を認識する
チューリング機械



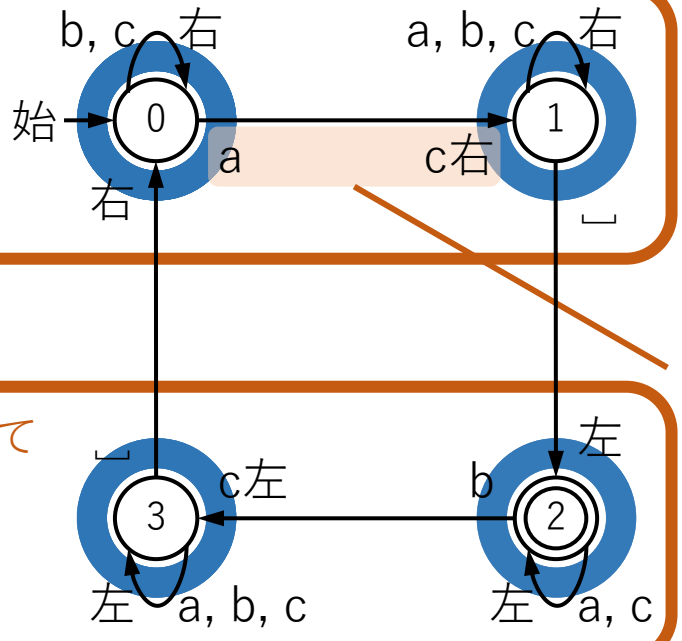
問題
MOREA

与えられた文字列に
a が b よりも多く現れるか答えよ



初めは状態 0
右に読み進め　最初の a を
c に書き換えて状態 1 になり
そのまま右端まで進む

右端に達すると状態 2 になって
左へ戻ってゆき　最初の b を
c に書き換えて状態 3 になり
そのまま左端まで進む



状態 0 で
文字 a を読むと
- 状態を 1 にし
- 文字 c を書込み
- 右へ行く

辿れる矢印が
ないときは**止**

読取りのみでは解けない問題を解けるようになった！

定義

言語 A を認識する（読取りのみの）有限状態機械が存在するとき
A は**正則**であるという

regular

定義

言語 A を認識するチューリング機械が存在するとき
A は**認識可能**である（$A \in \mathbf{CE}$）という

computably enumerable
(recursively enumerable)

でも文字列に関する問題しかないの？

入力を**符号化**する方法を決めた上で文字列に関する問題として扱います

問題 PRIME

0 1 … 9 からなる文字列

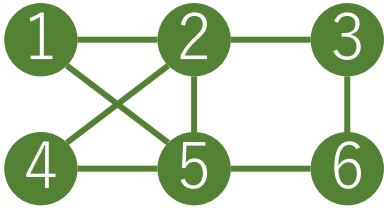
与えられた正整数（十進法で書く）が素数か

問題 HAMILTON

0 1 … 9 ( ) , からなる文字列

与えられたグラフ（次のような形式で書く）がハミルトン閉路をもつか

（全頂点を一度づつ通る辿り方）



符号化

6,(1,2),(1,5),(2,3),(2,4),
(2,5),(3,6),(4,5),(5,6)

※ それなりに常識的な符号化法なら計算できるかどうかの議論に支障ないことが多いので今後あまり明記しないこともある

チャーチとチューリングの定立

「計算できる」
(機械的な手順で解ける)
ボンヤリした概念

＝

(チューリング機械で)
認識可能
数学的にハッキリ定義した概念

チューリング機械ですべての「計算」が実現できているなんて本当？

幾つかの理由から　今では広く受入れられています

- 現実の計算に出て来そうな色々な手順は
  確かにチューリング機械で書けるようだ (経験上)
- 他の様々なやり方で定義された計算可能性の概念とも一致

チューリングの論文 (次頁)

A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 2, **42**: 230–265, 1936.

1. [Type (*a*)]. This argument is only an elaboration of the ideas of § 1.

Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent†. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

紙に文字を書きながら計算する様子を考えよう

文字は有限個と考えてよかろう

無限個使っても区別できなきゃ意味ないので

† If we regard a symbol as literally printed on a square we may suppose that the square is $0 \leqslant x \leqslant 1$, $0 \leqslant y \leqslant 1$. The symbol is defined as a set of points in this square, viz. the set occupied by printer's ink. If these sets are restricted to be measurable, we can define the "distance" between two symbols as the cost of transforming one symbol into the other if the cost of moving unit area of printer's ink unit distance is unity, and there is an infinite supply of ink at $x = 2$, $y = 0$. With this topology the symbols form a conditionally compact space.

17 or 999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 999999999999999 are the same.

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.

大きな数値を記号と考えてもやはり瞬時には区別できない

現在の文字と現在の状態から次の動きを決める

状態も有限個と考えてよかろう

一度に読み書きできる文字数も

チューリング機械と（適当な符号化の下で）等価な計算可能性の定義としては
例えば次の「再帰的函数」がある

定義

自然数の組から自然数への部分函数 $f\colon\subseteq \mathbf{N}^n \rightharpoonup \mathbf{N}$ $(n = 0, 1, 2, \ldots)$ が
再帰的 (recursive) であることを次で定める

- 次で定義される $\mathrm{zero}\colon \mathbf{N}^0 \to \mathbf{N}$ と $\mathrm{proj}_i^n\colon \mathbf{N}^n \to \mathbf{N}$ と $\mathrm{succ}\colon \mathbf{N} \to \mathbf{N}$ は再帰的
$$\mathrm{zero}(\ \ ) = 0 \qquad \mathrm{proj}_i^n(x_1, \ldots, x_n) = x_i \qquad \mathrm{succ}(x) = x + 1)$$
- $g\colon\subseteq \mathbf{N}^n \rightharpoonup \mathbf{N}$ と $h_1, \ldots, h_n\colon\subseteq \mathbf{N}^m \rightharpoonup \mathbf{N}$ とが再帰的ならば
次で定義される $f\colon\subseteq \mathbf{N}^m \rightharpoonup \mathbf{N}$ も再帰的
$$f(x) = g\big(h_1(x), \ldots, h_n(x)\big)$$
- $g\colon\subseteq \mathbf{N}^n \rightharpoonup \mathbf{N}$ と $h\colon\subseteq \mathbf{N}^{n+2} \rightharpoonup \mathbf{N}$ が再帰的ならば
次で定義される $f\colon\subseteq \mathbf{N}^{n+1} \rightharpoonup \mathbf{N}$ も再帰的
$$f(0, y) = g(y)$$
$$f(x + 1, y) = h\big(x, y, f(x, y)\big)$$
- $g\colon \mathbf{N}^{n+1} \rightharpoonup \mathbf{N}$ が再帰的ならば　次で定義される $f\colon\subseteq \mathbf{N}^n \to \mathbf{N}$ も再帰的
$$f(x) = \min\{\, y \in \mathbf{N} \mid g(x, y) = 0 \,\}$$

但し各等式の左辺は「右辺が順次定義されるとき定義される」と解釈する
(例えば最後のものでは $g(x, 0), \ldots, g(x, y - 1)$ がみな定義されて 0 でなく
かつ $g(x, 0) = 0$ であるような y が存在するときに $f(x)$ が定義される)

機械は文字列で表せる

万能機械

プログラマ

チューリング機械は（有限の）文字列で記述できる

遷移規則 $\delta(q,\sigma)=(q',\sigma',d)$ が有限個あるだけ

その文字列（プログラム）（算譜）を機械と呼ぶと思ってもよい

このお蔭で　各機械を実際に建造せずとも
算譜を読込むことで同じ機能を実現できる

ε
0
1
00
01
10
11
000
001
010
011
100
101
110
111
0000
0001
0010
0011
0100
0101

すべての機械はこのどこかに現れる

問題 EVAL

入力　二つの文字列の組 (M,x)

答　機械 M は入力 x を受理するか

「算譜を実行する」問題

定理

EVAL $\in$ **CE**

EVAL を認識するチューリング機械を「万能チューリング機械」という

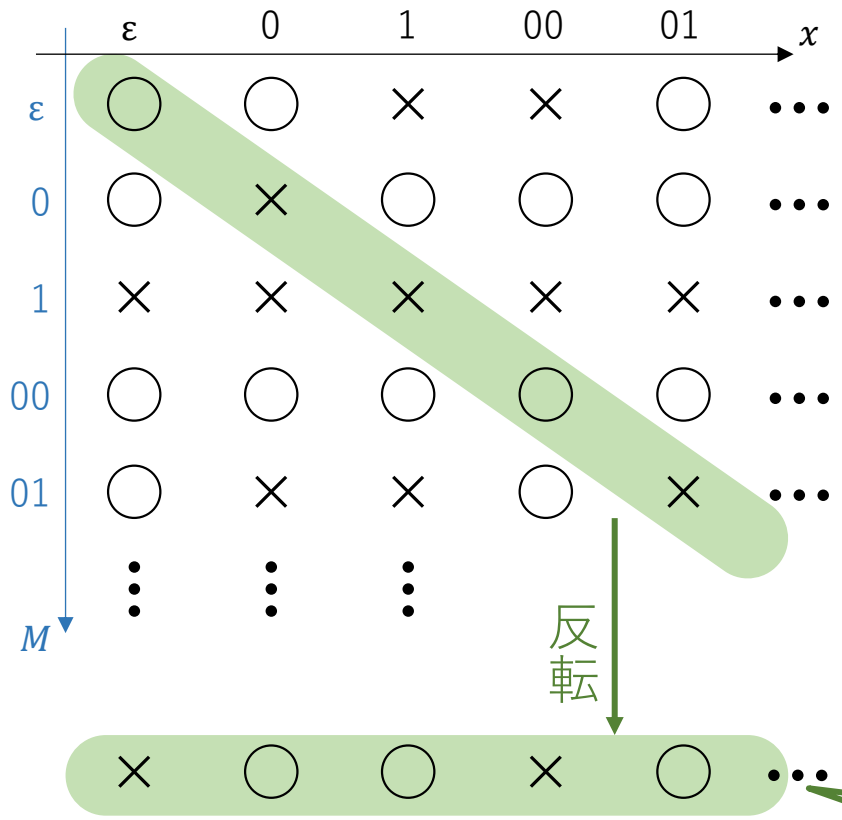
証明（？）「チャーチとチューリングの定立」に頼ると……

先程やったように
チューリング機械（を表す図）を見ながらそれを実行することは
機械的作業である（ので同じことをチューリング機械でできる）

しかし　機械が有限の文字列で表せることから

認識可能でない言語

を構成することもできてしまう



対角線論法

図のように

文字列に対応づけて無限に

○と×を横に並べたものを**○×行**と呼び

○×行を縦に並べたものを**○×表**と呼ぶ

定理

どんな○×表に対しても
それに現れない○×行が存在する

∵　左図のように　対角線上の内容と喰い違うように定義すればよい

各行を機械の動作と考えると……

チューリング機械 M が入力 x を受理するか否かを第 (M, x) 成分に○×で記した表にこの定理を適用

どの機械でも認識されない言語

今の議論で結局　何という問題が認識不能と示されたのか

問題 EVAL	入力	二つの文字列の組 (M, x)	**認識可能**
	答	機械 M は入力 x を受理するか	
問題 $\overline{\text{EVAL}}$ (EVAL の補集合)	入力	二つの文字列の組 (M, x)	**認識可能でない**
	答	機械 M は入力 x を受理**しない**か	
対角線論法で作った問題	入力	文字列 x	**認識可能でない**
	答	機械 x は入力 x を受理**しない**か	

EVAL は認識可能 (**CE** に属する) だが
判定可能 ($\mathbf{C = CE \cap coCE}$ に属する) ではない

任意の入力 x に対し
- もし $x \in$ EVAL ならば x を受理 (して停止)
- もし $x \notin$ EVAL ならば x を**受理せずに停止**すること

M に x を入力した計算が
停止しないことを
有限の時間で確実に
予言する方法はない

問題 SR

入力　書換え規則の（有限）集合 R と文字列 $w \in \Sigma^*$

R の各規則は $u \to v$ という形（$u, v \in \Sigma^*$）

文字列の一部を u から v に書換えることができるという意味

つまり $xuy \Rightarrow_R xvy$ とできる

答　R による書換えを次々と w に施して ε（空文字列）にできるか

$w \Rightarrow_R^* \varepsilon$ と書くことにする

例 1

入力

$$R \begin{cases} \text{aa} \to \text{bbb} \\ \text{aba} \to \text{a} \\ \text{bb} \to \varepsilon \end{cases}$$

$w = \text{aababab}$

答　○（受理）

（aababab $\Rightarrow_R$ aabab $\Rightarrow_R$ aab $\Rightarrow_R$ bbbb $\Rightarrow_R$ bb $\Rightarrow_R \varepsilon$ とできるので）

例 2

入力

$$R \begin{cases} \text{aa} \to \text{bab} \\ \text{aba} \to \text{a} \\ \text{bb} \to \varepsilon \end{cases}$$

$w = \text{aababab}$

答　×（不受理）

（　？　ので）

問題 SR

入力　書換え規則の（有限）集合 R と文字列 $w \in \Sigma^*$

R の各規則は $u \to v$ という形（$u, v \in \Sigma^*$）

文字列の一部を u から v に書換えることができるという意味

つまり $xuy \Rightarrow_R xvy$ とできる

答　R による書換えを次々と w に施して ε（空文字列）にできるか

$w \Rightarrow_R^* \varepsilon$ と書くことにする

例 1

入力

$$R \begin{cases} \text{aa} \to \text{bbb} \\ \text{aba} \to \text{a} \\ \text{bb} \to \varepsilon \end{cases}$$

$w = \text{aababab}$

答　○（受理）

（aababab $\Rightarrow_R$ aabab $\Rightarrow_R$ aab $\Rightarrow_R$ bbbb $\Rightarrow_R$ bb $\Rightarrow_R \varepsilon$ とできるので）

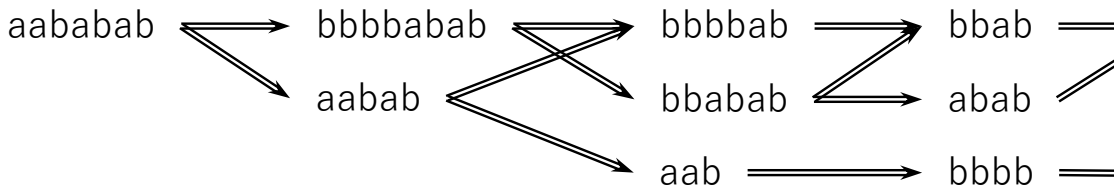
定理

SR ∈ **CE**

書換え 1 回で作れる文字列をすべて列挙（ε があれば受理）

書換え 2 回で作れる文字列をすべて列挙（ε があれば受理）

……と順に調べてゆけばよい



（この方法では $w \Rightarrow_R^* \varepsilon$ でないときは計算が停止しない）

実は SR は判定可能でないことが後で判る

その前にまず　二つの似た問題のどちらが
より判定可能でありそうか比べる論法 (**帰着**) について考えよう

問題 SR

入力　書換え規則の集合 R と文字列 w

答　R による書換えを次々と w に施して空文字列にできるか

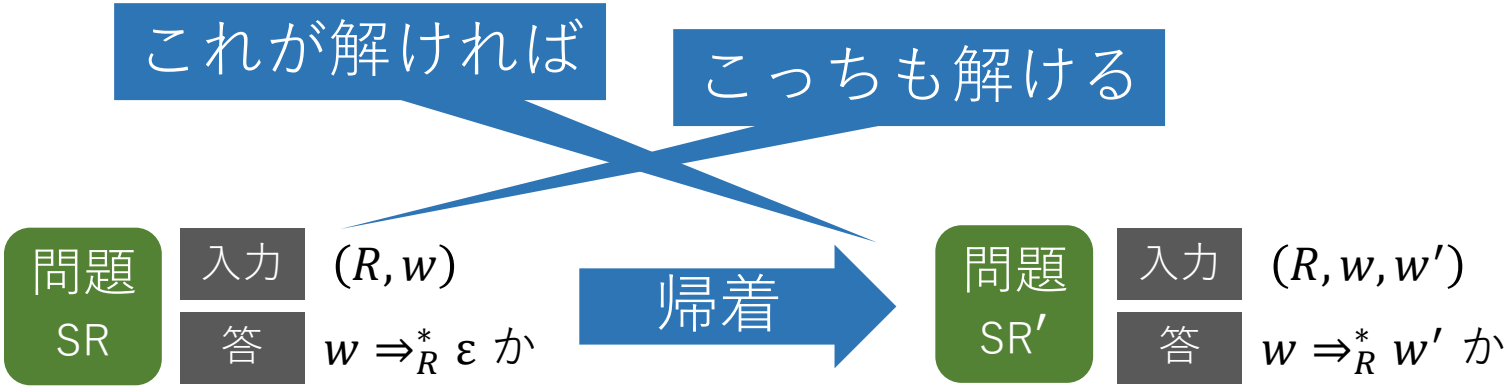
問題 SR′

入力　書換え規則の集合 R と文字列 w, w'

答　R による書換えを次々と w に施して w' にできるか

どちらが
より難しい？

問題の難しさの比較 (**帰着**)



入力

(R, w)

が SR に属するか知りたい

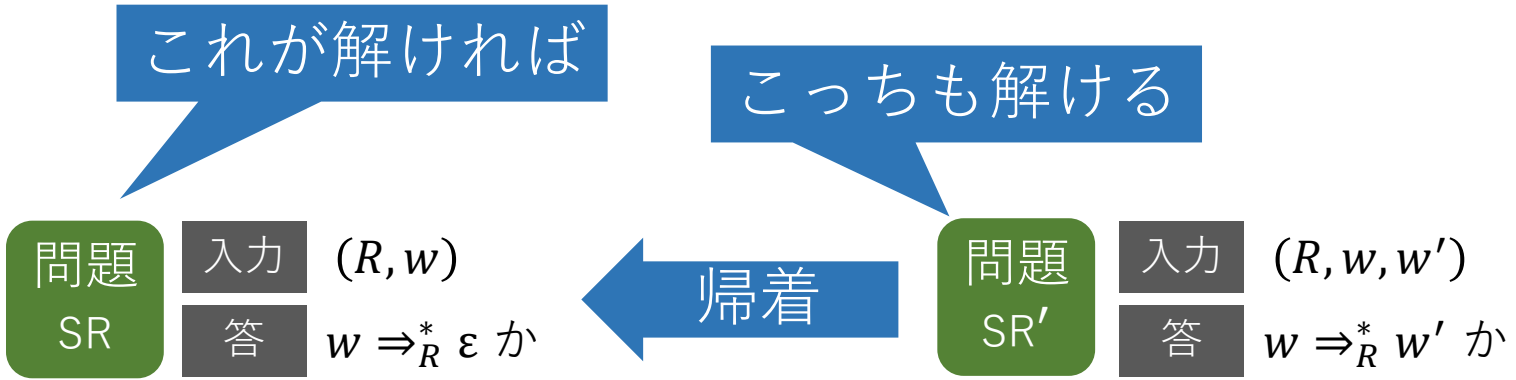
そのためには

(R, w, ε)

が SR′ に属するか調べれば良い

$$(R, w) \in \mathrm{SR} \quad \Leftrightarrow \quad (R, w, \varepsilon) \in \mathrm{SR}'$$

問題の難しさの比較（**帰着**）



入力

$(R \cup \{\triangleright w' \triangleleft \to \varepsilon\}, \triangleright w \triangleleft)$ ⟵ (R, w, w')

（$\triangleright$ と $\triangleleft$ は新たな文字）

が SR′ に属するか知りたい

$$(R \cup \{\triangleright w' \triangleleft \to \varepsilon\}, \triangleright w \triangleleft) \in \mathrm{SR} \iff (R, w, w') \in \mathrm{SR}'$$

二つの問題は（判定可能かどうかに関しては）
「同じ難しさ」であることが判った！

問題の難しさの比較 (**帰着**)

入力　書換え規則の集合 R と文字列 w

答　R による書換えを次々と w に施して空文字列にできるか

(前頁)

入力　書換え規則の集合 R と文字列 w, w'

答　R による書換えを次々と w に施して w' にできるか

問 2　この帰着 (SR″ も同じ難しさであること) を示して下さい

入力　書換え規則の集合 R と文字列 w, w''

答　R による書換えを次々と w に施して
w'' が現れる文字列にできるか

定理

SR $\notin \mathbf{C}$ (すなわち $\overline{\text{SR}} \notin \mathbf{CE}$)

これが判定不能と判っているので

これも

これも判定不能

∵ 右図の帰着による

(M, x) ⋯▸ $(R, \triangleright q_{始} x \triangleleft, ㊎)$

但し R は　M の各状態 $q \in Q$ と文字 $\sigma \in \Gamma$ について次の規則を加えて作る

- もし $\delta(q,\sigma) = (q', \sigma', ㊧)$ ならば　任意の $\tau \in \Gamma$ に対し規則 $\tau q \sigma \to q' \tau \sigma'$
- もし $\delta(q,\sigma) = (q', \sigma', ㊨)$ ならば　規則 $q\sigma \to \sigma' q'$
- もし $\delta(q,\sigma) =$ 止 かつ $q \in Q_{受理}$ ならば　規則 $q\sigma \to ㊎$

また　規則 $\triangleright \to \triangleright \sqcup$ および規則 $\triangleleft \to \sqcup \triangleleft$ も R に含める　すると

$(M, x) \in$ EVAL ⇔ 状況 $q_{始} x$ から遷移 M を辿ってゆくと
$\delta(q,\sigma) =$ 止 かつ $q \in Q_{受理}$ なる $q\sigma$ が現れる状況に到達

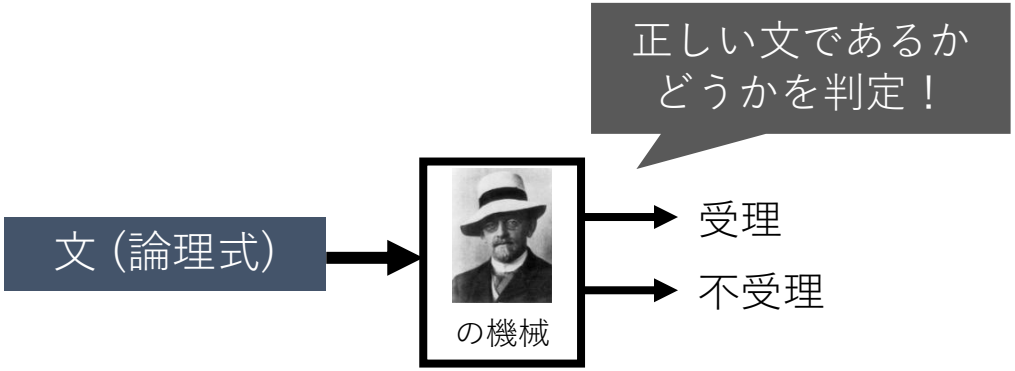
⇔ 文字列 $\triangleright q_{始} x \triangleleft$ に R の規則を施して㊎を含む文字列に辿り着ける

(すなわち $(R, \triangleright q_{始} x \triangleleft, ㊎) \in$ SR″)

ヒルベルトの判定問題

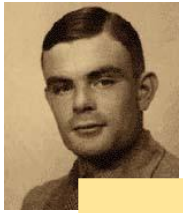
一階述語論理で書かれた文が与えられたとき
それが正しい (＝証明可能) か否かを判定する
ような機械的な手順はあるか？(1928)

A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 2, **42**: 230–265, 1936.



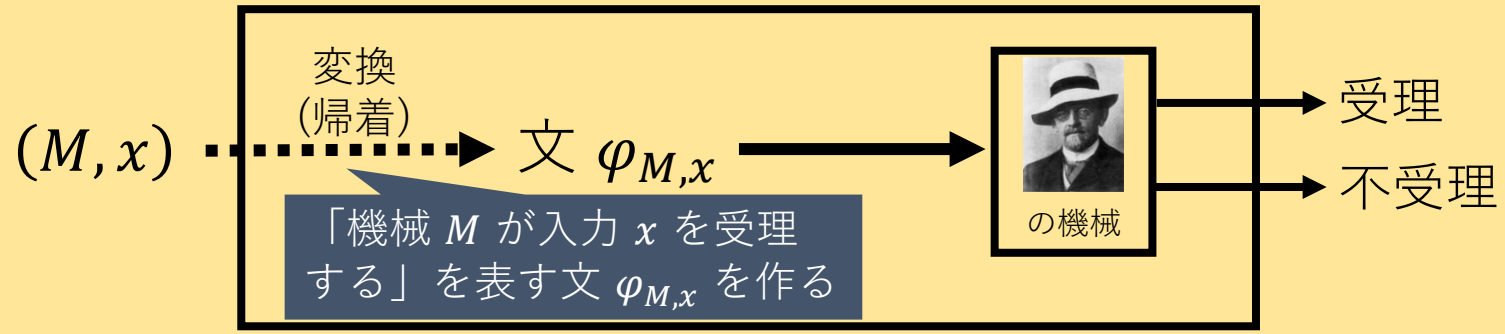
ヒルベルトの判定問題

一階述語論理で書かれた文が与えられたとき
それが正しい (＝証明可能) か否かを判定する
ような機械的な手順はあるか？(1928)



A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Soci* **42**: 230–265, 1936.

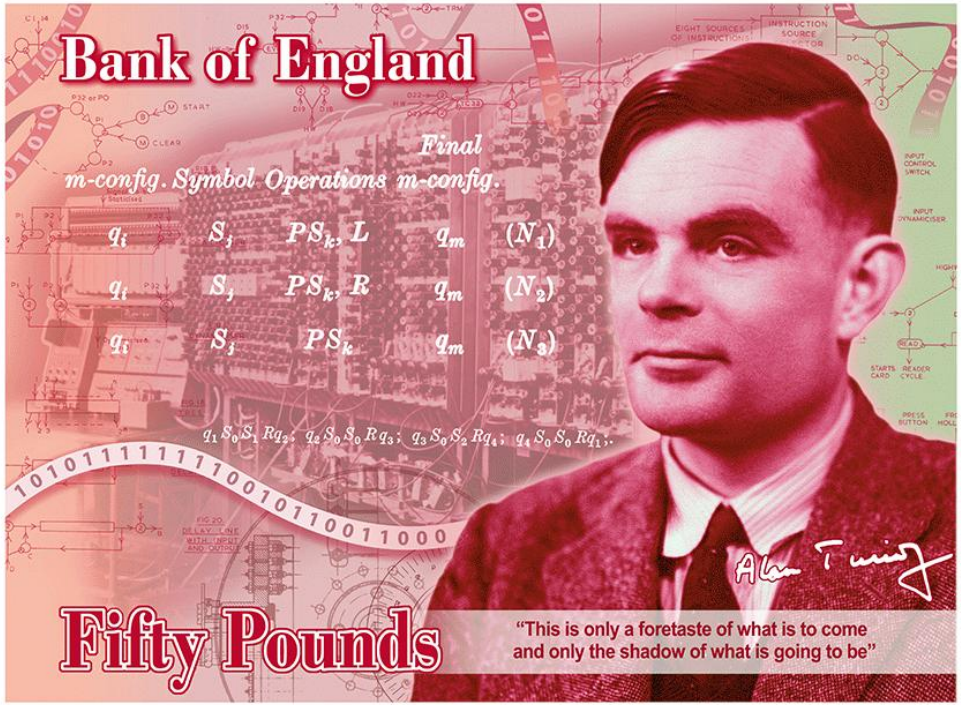
もしあるとすると先程の EVAL が判定できてしまうので、無い



五十ポンド紙幣（２０２１年～）

BANK OF ENGLAND

Alan Turing Banknote Concept

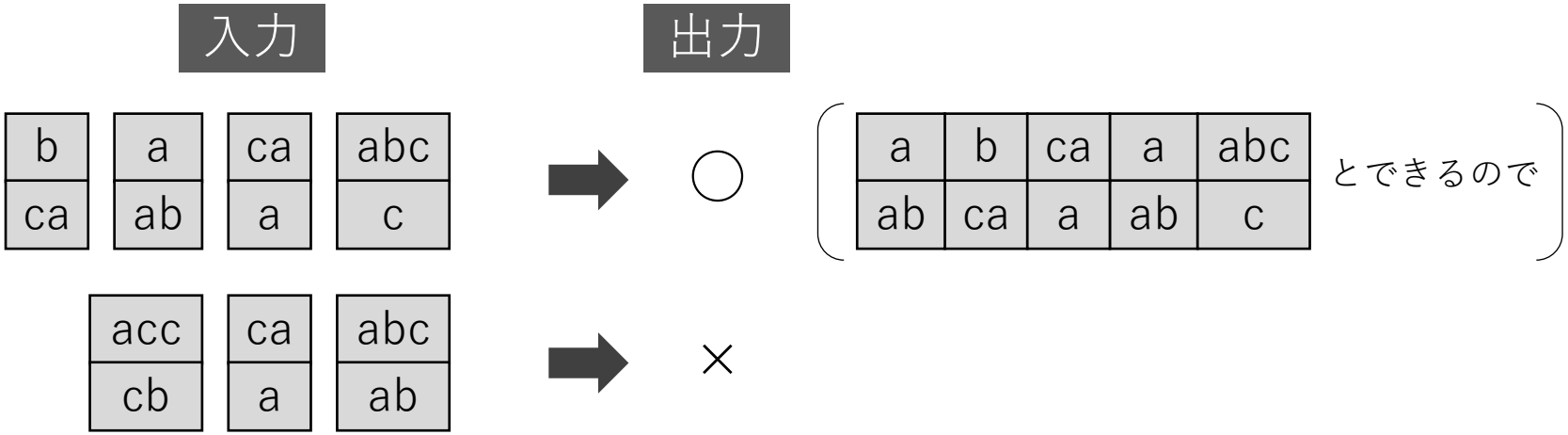


有名な判定不能問題の例

問題（ポストの対応問題）

入力　何枚かの上下に文字列の書かれた札（各種類の札がいくらでもある）

答　これらを横に有限枚並べて
上段と下段の文字列を一致させることができるか？



有名な判定不能問題の例

問題

入力　何枚かの上下に文字列の書かれた札 (各種類の札がいくらでもある)

答　これらを横に有限枚並べて
上段と下段の文字列を一致させることができるか？

(ポストの対応問題)

問題

入力　整数係数 (多変数) 多項式 p

答　$p = 0$ は整数解をもつか

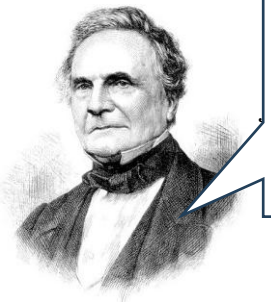
(ヒルベルトの第10問題)

問題

入力　いくつかの整数 3×3 行列

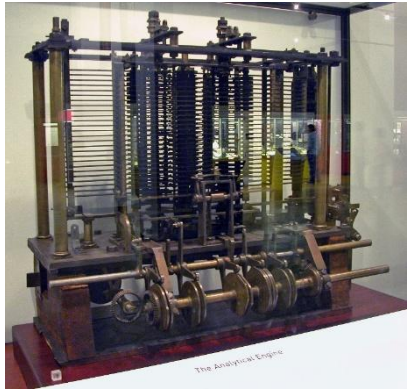
答　これらの行列をうまく掛け合わせて 0 にできるか
(同じ行列を何度どういう順で使ってもよい)

計算量 (Computational Complexity)



解析機関は、実現すれば必ずや将来の科学に重要なものとなろう。この機械を用いて結果を得ようとするとき問われるのは、**いかなる手順で計算すれば最も短時間で結果に辿り着けるか**である。

— *Charles Babbage (1864)*



解析機関 (Analytical Engine)

判定可能な問題の間でも「どれほど効率よく判定可能か」による違いが重要

「解析機関」『ウィキペディア日本語版』https://ja.wikipedia.org/w/index.php?title=%E8%A7%A3%E6%9E%90%E6%A9%9F%E9%96%A2&oldid=82653674

計算量の考え方 (原則)　1950～60年代

- チューリング機械での計算にかかる
  時間 (遷移の回数) や空間 (訪れる欄の数) を考える
  現実にかかる時間や空間をよく表している

- それが入力の長さに応じてどう変るか函数として表す
  「長さ n の入力なら必ず時間 (や空間) が $T(n)$ 以内で済む」
  ような函数 T が計算量の尺度 (最悪時評価)

- その函数の増大の速さに着目
  特に n の多項式以内か否かが重要

定義

機械 M が**多項式時間**であるとは　或る多項式 $p\colon \mathbf{N} \to \mathbf{N}$ が存在し
任意の長さ n の任意の入力に対する M の計算が
時間 $p(n)$ 以内に終る（遷移 $p(n)$ 回以下で停止する）ことをいう

言語 A を認識する多項式時間の機械 M が存在するとき
A は**多項式時間認識可能**である（$A \in \mathbf{P}$）という

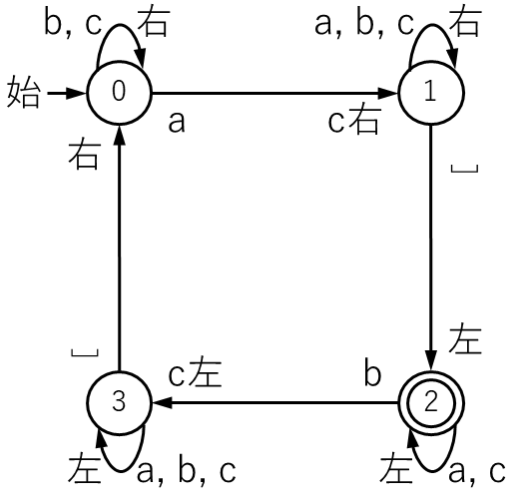
n の多項式以内である例

$$n \qquad n\sqrt{n} \qquad n^2 \log n \qquad 5n^3 + 4n$$

n の多項式以内でない例

$$2^n \qquad 2^{2^n} \qquad 1.05^n \qquad n! \qquad n^{\log n}$$

例　先程のこの機械は
時間 $(n+1)^2$ 以内に停止するので
MOREA $\in \mathbf{P}$



定義

機械 M が**多項式時間**であるとは　或る多項式 $p\colon \mathbf{N} \to \mathbf{N}$ が存在し
任意の長さ n の任意の入力に対する M の計算が
時間 $p(n)$ 以内に終る（遷移 $p(n)$ 回以下で停止する）ことをいう

言語 A を認識する多項式時間の機械 M が存在するとき
A は**多項式時間認識可能**である（$A \in \mathbf{P}$）という

「認識可能」の定義では $x \notin A$ のときは停止しないことを許していたのだから
それに合せて「多項式時間認識」の定義は　長さ n の入力 x について

- $x \in A$ のとき M は x を時間 $p(n)$ 以内に受理する
- $x \notin A$ のとき M は x を時間 $p(n)$ 以内に受理しない

とすべきでは？

そう定義しても
P の意味は変りません

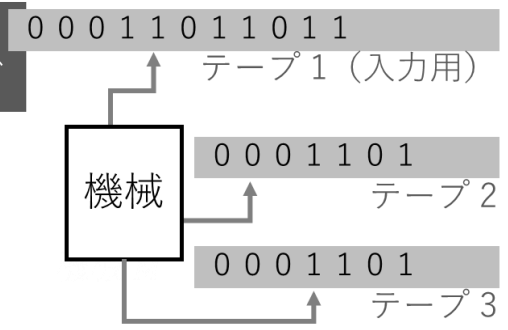
受理せずに時間 $p(n)$ が経過したら
不受理と確定できるので

多項式時間では「認識可能」と「判定可能」の違いはない
（多項式時間認識可能な言語は補集合も多項式時間認識可能）

チャーチとチューリングの定立 (続)

「現実的な手間で計算できる」 ＝ (チューリング機械で) 多項式時間認識可能

テープの形や
読取り部の動き方など



チューリング機械の定義の細部は
計算法が多項式時間であるかないかに
影響を与えない

（「$O(n^3)$ であるかないか」など細かい量には
それなりに影響がある）

今後は一々チューリング機械を考えず　計算手順が解り易いように説明する

時間＝「基本的な操作の回数」と大雑把に考えてよい
- 四則演算やビットの操作など
- 機械語での命令数

なぜ多項式時間か否かが重要か

■ 入力が大きくなると手間が大違い

入力長 $n=$	10	30	50	100	1000	1万	100万	1億
$\sqrt{n}$	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内
n	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内	1秒以内	1秒	2分
n^2	1秒以内	1秒以内	1秒以内	1秒以内	1秒	2分	12日	3百年
n^3	1秒以内	1秒以内	1秒以内	1秒	17分	12日	3万年	3百億年
2^n	1秒以内	18分	36年	4京年				
10^n	17分	3京年						
$n!$	3.6秒	800京年						

多項式時間

指数時間

1秒に100万回の処理が
できるとしたときにかかる時間

実際には　多項式時間の中での速さの差
(n^2 と n^3 の違いなど) も勿論重要なのですが
この講義では「多項式時間であるかないか」という
より大きな違いに着目します

なぜ多項式時間か否かが重要か

■ 指数個 (以上) の組合せから何かを探す場面は多い (組合せ爆発)

問題 PRIME 与えられた正整数 (十進法で n 桁) が素数か判定

それ未満の数 (10^n 個ほどある) で割り切れるかすべて調べれば判る

つまり　整数 x を文字列 0^x で表すという「非効率」な符号法で入力を受付ける素数判定問題なら自明に **P**

それよりも劇的に速く　n の多項式時間で判定する方法が存在 [AKS05]

問題 HAMILTON 与えられたグラフ (頂点数 n) がハミルトン閉路をもつか判定
(全頂点を一度づつ通る辿り方)

頂点の並べ順 ($n!$ 通り) をすべて調べれば判る

n の多項式時間で判定する方法があるかどうか不明 (多分なさそう)

問題 SAT 与えられた命題論理式 (変数が n 個) が充足可能か判定

真理値割当 (2^n 通り) をすべて調べれば判る

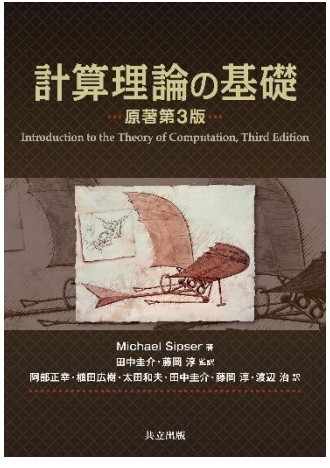
n の多項式時間で判定する方法があるかどうか不明 (多分なさそう)

[AKS05] M. Agrawal, N. Kayal and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160: 781–793, 2004.

まとめ　計算可能性理論

- **問題**（**言語**）　無限個の入力に正解を定めたもの
- **機械**（**算譜**）　有限に記述される計算手順
- (チューリング機械で) 認識可能 ≒「計算できる」
- EVAL は認識可能 (入力された算譜を実行する**万能機械**の存在)
- しかし対角線論法により　EVAL は判定不能
- そこから**様々な問題の判定不能性**が帰着により示される
- このような不可能性の証明は
  計算の概念を明確にしてこそ可能になった
- 計算量の制限　**多項式時間**など

M・シプサ著、田中・藤岡監訳『計算理論の基礎 原著第三版』共立出版（令和5年）



岩間一雄『アルゴリズム理論入門』朝倉書店（平成26年）