# 「計算」の定式化

**計算機に於ける「計算」の各ステップ**
**(= 命令の実行) は、**

- **外部からの入力**
- **内部状態 (メモリ・レジスタ) の現在の値**

**に従って、**

- **外部への出力**
- **内部状態 (メモリ・レジスタ) の値の変更**

**を行なうこと**

# 計算の理論

**プログラム内蔵方式 (von Neumann 型) では、**

**プログラム・データを区別なくメモリ上に置くが、**

**プログラムとデータとは、やはり本質的に違う**

- **プログラム: 一つの問題では固定**
- **データ: 可変な入力**

⇓

**どんな (有効な) データ (入力) が来ても、**
**所定の出力を返すことが要請される**

# 計算の理論

**プログラム内蔵方式 (von Neumann 型) では、**

**プログラム・データを区別なくメモリ上に置くが、**

**プログラムとデータとは、やはり本質的に違う**

- **プログラム: 一つの問題では固定**
- **データ: 可変な入力**

⇓

**どんな (有効な) データ (入力) が来ても、**
**所定の出力を返すことが要請される**

# 計算の理論

**プログラム内蔵方式 (von Neumann 型) では、**

**プログラム・データを区別なくメモリ上に置くが、**

**プログラムとデータとは、やはり本質的に違う**

- **プログラム: 一つの問題では固定**
- **データ: 可変な入力**

**どんな (有効な) データ (入力) が来ても、**
**所定の出力を返すことが要請される**

# 計算の理論

**或る問題の「計算が可能」**

$\Updownarrow$

**その計算を行なうプログラムが存在**

$\Downarrow$

**計算機の機能 ( ＝「計算」のモデル) を決めて議論**

$\longrightarrow$ **代表的な「計算のモデル」を幾つか紹介**

# 計算の理論

**或る問題の「計算が可能」**

$\Updownarrow$

**その計算を行なうプログラムが存在**

$\Downarrow$

**計算機の機能 ( $=$「計算」のモデル) を決めて議論**

$\longrightarrow$ **代表的な「計算のモデル」を幾つか紹介**

# 計算の理論

**或る問題の「計算が可能」**

$\Updownarrow$

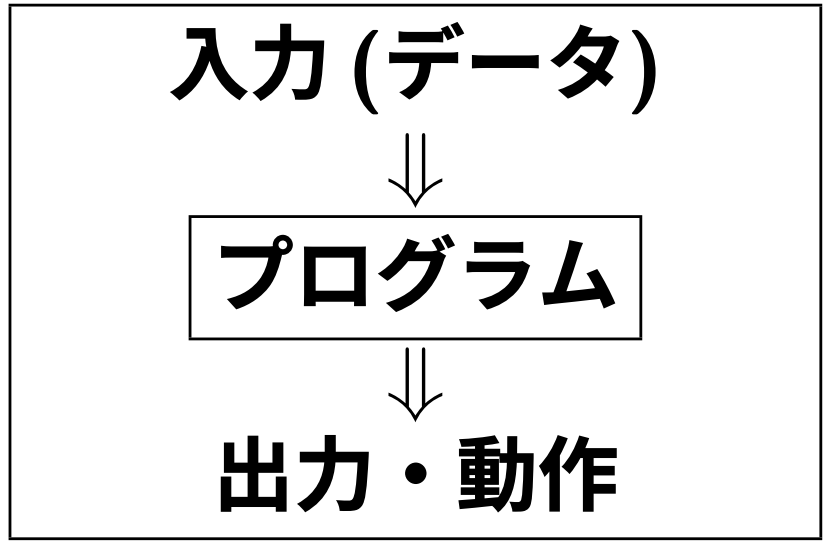
**その計算を行なうプログラムが存在**

$\Downarrow$

**計算機の機能 ( $=$「計算」のモデル) を決めて議論**

$\longrightarrow$ **代表的な「計算のモデル」を幾つか紹介**

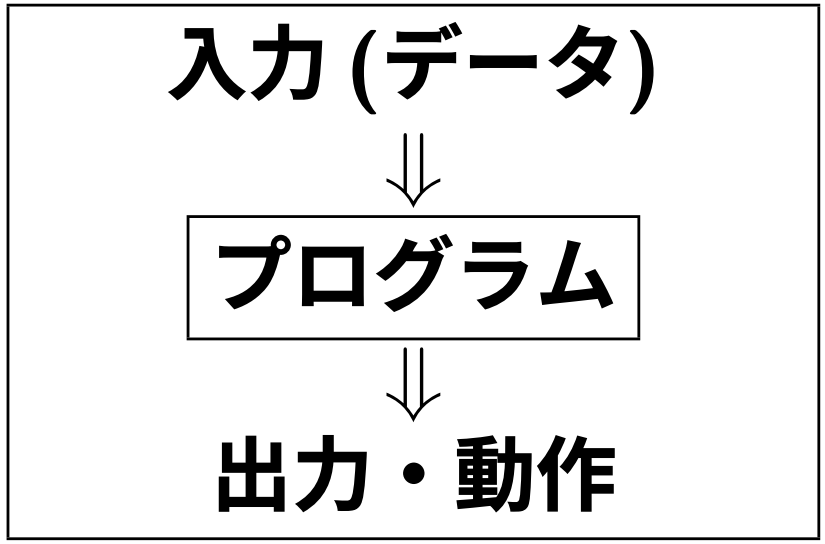
# 問題を「計算する」とは




原理・理論を考える際には、
出力は最も単純に「0 か 1 か」とする

- 0 : 拒否 (reject)
- 1 : 受理 (accept)

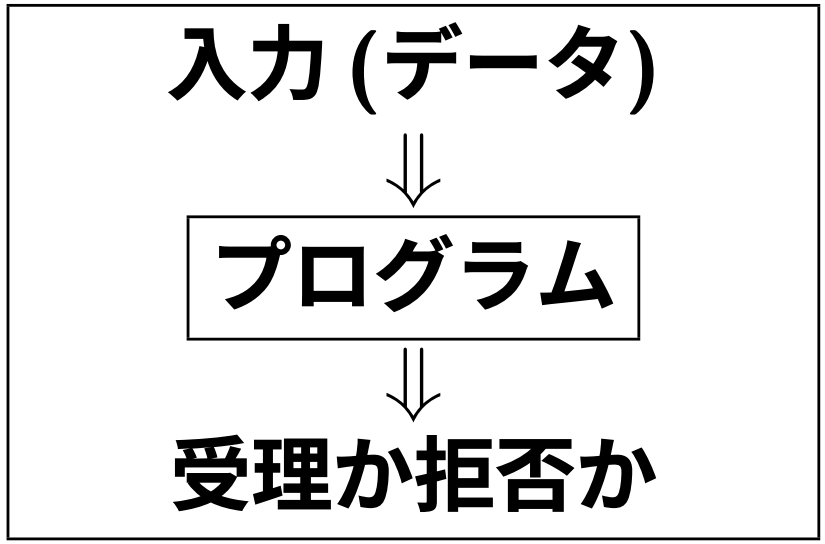
# 問題を「計算する」とは




**原理・理論を考える際には、**
**出力は最も単純に「0 か 1 か」とする**

- 0 : **拒否** (reject)
- 1 : **受理** (accept)

# 「問題」とは




**解くべき「問題」: 入力を受理する条件**

**問題を「解く」: 入力が条件を満たすか判定する**

# 「問題」の例

**入力の範囲：文字 $\mathtt{a}, \mathtt{b}$ から成る文字列**

**「問題」：入力を受理する条件**

- $\mathtt{a}$ と $\mathtt{b}$ との個数が同じ
- $\mathtt{a}$ が幾つか続いた後に $\mathtt{b}$ が幾つか続いたもの
- $\mathtt{a}$ で始まり $\mathtt{a}, \mathtt{b}$ が交互に並んで $\mathtt{b}$ で終わる
- 同じ文字列 $2$ 回の繰返しから成る
- 回文 (palindrome)

**などなど**

# 「問題」とは

**それぞれの「問題」に対し、**
**定められた計算モデルで、**
**受理／拒否判定が可能 (問題が解ける) か ？**

受理される文字列が
「文法に適っている」文字列だと思えば、

「問題」とは「文法 (言語)」である

「文法に適っている」かどうかの判定
. . . 「構文解析 (syntactic analysis)」

# 「問題」とは

**それぞれの「問題」に対し、**
**定められた計算モデルで、**
**受理／拒否判定が可能 (問題が解ける) か ？**

**受理される文字列が**
**「文法に適っている」文字列だと思えば、**

**「問題」とは「文法 (言語)」である**

「文法に適っている」かどうかの判定
. . . 「構文解析 (syntactic analysis)」

# 「問題」とは

それぞれの「問題」に対し、
定められた計算モデルで、
受理／拒否判定が可能 (問題が解ける) か ？

受理される文字列が
「文法に適っている」文字列だと思えば、

「問題」とは「文法 (言語)」である

「文法に適っている」かどうかの判定
. . . 「構文解析 (syntactic analysis)」

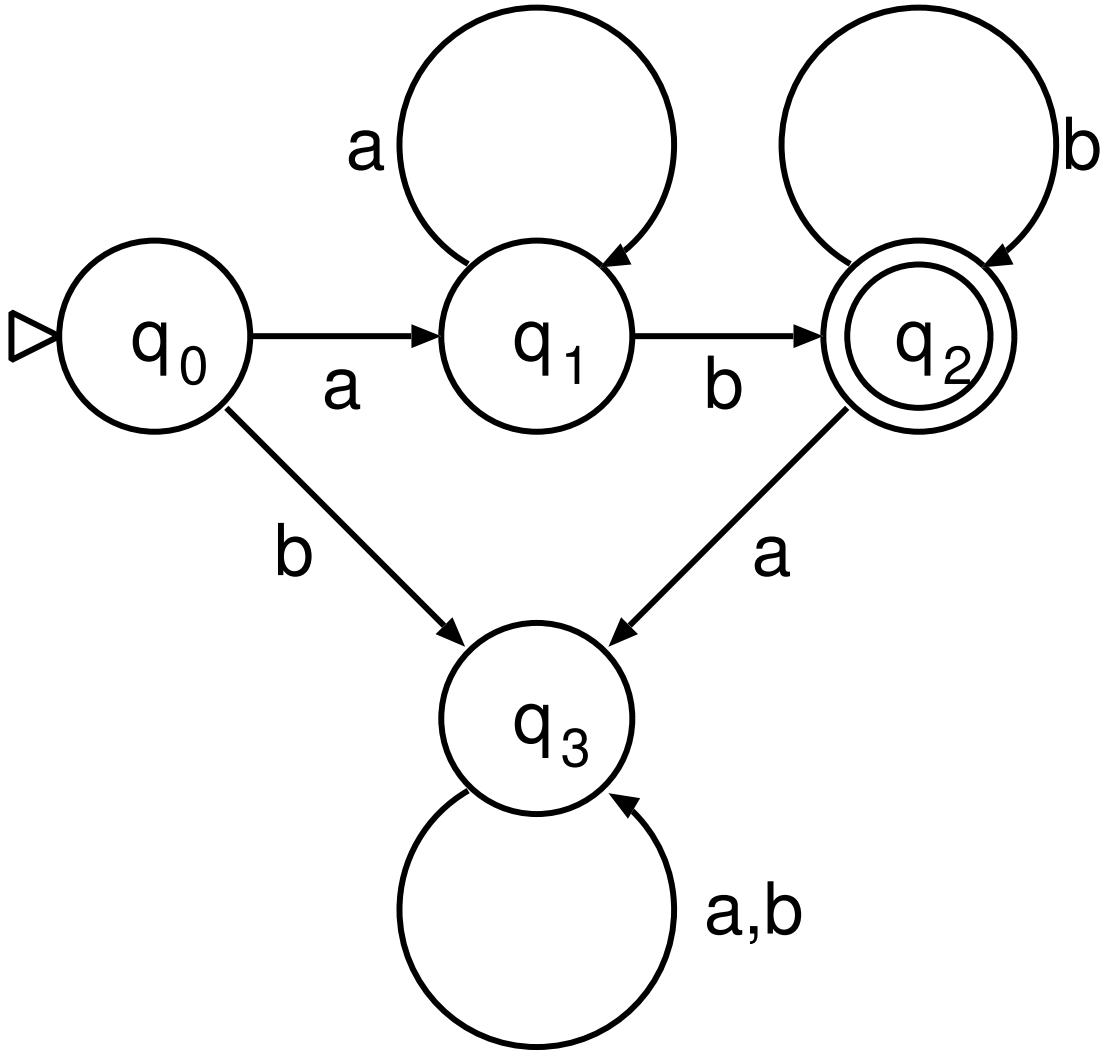
# 代表的な計算モデル

- 有限オートマトン (有限状態機械)
- プッシュダウンオートマトン
- チューリングマシン

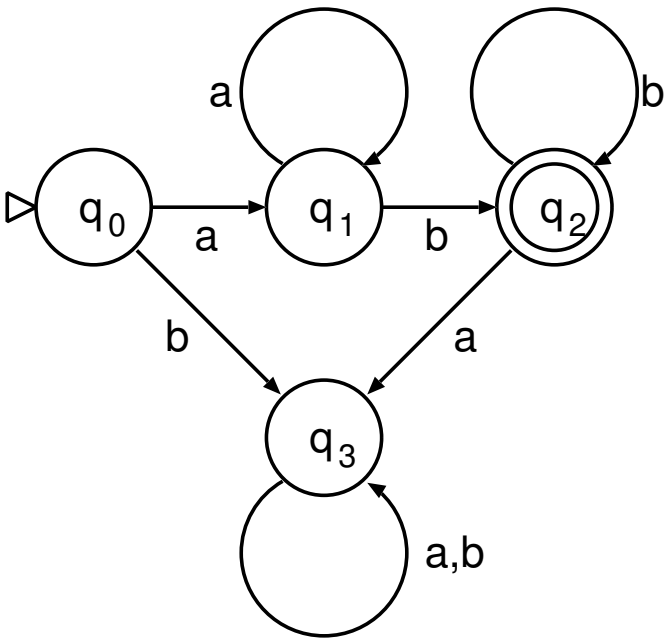
# 代表的な計算モデル

- **有限オートマトン (有限状態機械)**
- **プッシュダウンオートマトン**
- **チューリングマシン**

# 有限オートマトンの例 (状態遷移図による表示)



# 有限オートマトンの動作



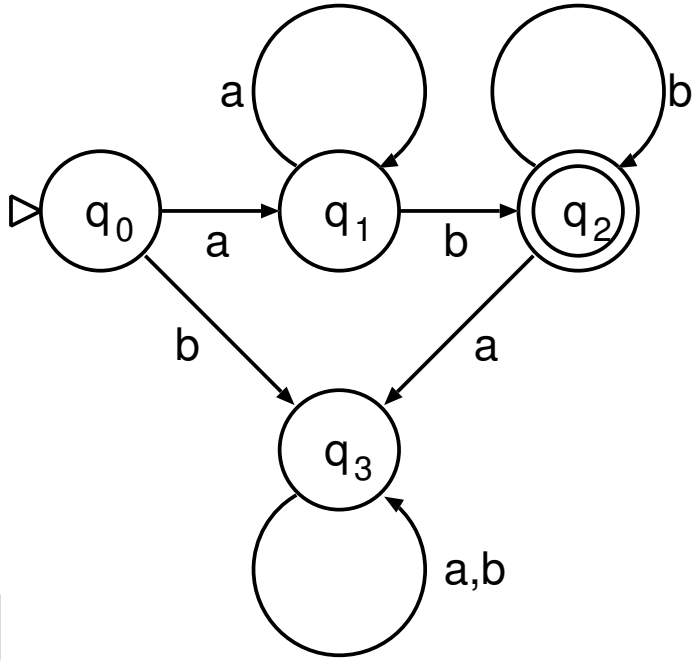

- 有限個の内部状態を持つ
- 有限個の文字から成る有限長の文字列を入力として受けて動作する
- 初期状態が定められている
- 入力を１文字読み、その文字と今の内部状態とに従って、次の内部状態に移る
- 入力を読み終わったときの内部状態によって受理／拒否が決まる

# 有限オートマトンの形式的定義

$$M = (Q, \Sigma, \delta, s, F)$$

ここに、

- $Q$ : 有限集合 $\cdots$ **状態**の集合
- $\Sigma$ : 有限集合 $\cdots$ 入力文字の集合: **"alphabet"**
- $\delta : Q \times \Sigma \to Q$ : **遷移関数**
- $s \in Q$ $\cdots$ **初期状態**
- $F \subset Q$ $\cdots$ **受理状態**の集合




**先の例では、**

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- $\delta : Q \times \Sigma \to Q$ :

| | $q_0$ | $q_1$ | $q_2$ | $q_3$ |
|---|---|---|---|---|
| a | $q_1$ | $q_1$ | $q_3$ | $q_3$ |
| b | $q_3$ | $q_2$ | $q_2$ | $q_3$ |

- $s = q_0 \in Q$
- $F = \{q_2\} \subset Q$

## 語・言語

$\Sigma$ : 入力文字の有限集合 $\cdots$ alphabet

入力は $\Sigma$ の元の有限列 (語, word)

$$w = a_1 a_2 \cdots a_n \ (a_i \in \Sigma)$$

その全体 $\Sigma^*$

$$\Sigma^* := \bigcup_{n=0}^{\infty} \Sigma^n \qquad (\Sigma^0 = \{\varepsilon\} : \text{空列})$$

言語 (language) : $\Sigma^*$ の部分集合

言語 $A \subset \Sigma^*$ に属する語 $w \in A$

$\cdots$ 言語 $A$ に於いて “文法に適っている”

## 語・言語

$\Sigma$ : 入力文字の有限集合 $\cdots$ alphabet

入力は $\Sigma$ の元の有限列 (**語**, **word**)

$$w = a_1 a_2 \cdots a_n \ (a_i \in \Sigma)$$

その全体 $\Sigma^*$

$$\Sigma^* := \bigcup_{n=0}^{\infty} \Sigma^n \qquad (\Sigma^0 = \{\varepsilon\} : \text{空列})$$

**言語** (language) : $\Sigma^*$ の部分集合

言語 $A \subset \Sigma^*$ に属する語 $w \in A$

$\cdots$ 言語 $A$ に於いて “文法に適っている”

# 有限オートマトンによる語の受理

**有限オートマトン** $M = (Q, \Sigma, \delta, s, F)$ **が**
**語** $w = a_1 a_2 \cdots a_n$ **を受理 (accept) する**

$$\Updownarrow$$

$\exists r_0, r_1, \ldots, r_n \in Q$ :

- $r_0 = s$
- $\delta(r_{i-1}, a_i) = r_i$ $(i = 1, \ldots, n)$
- $r_n \in F$

$L(M)$ : $M$ **が受理する語の全体** $\subset \Sigma^*$
$\cdots$ $M$ **が認識 (recognize) する言語**

$M$ **は言語** $L(M)$ **の "文法" で、**
$M$ **が受理する語は "文法に適っている"**

# 有限オートマトンによる語の受理

**有限オートマトン** $M = (Q, \Sigma, \delta, s, F)$ **が**
**語** $w = a_1 a_2 \cdots a_n$ **を受理 (accept) する**

$$\Updownarrow$$

$\exists r_0, r_1, \ldots, r_n \in Q$ :

- $r_0 = s$
- $\delta(r_{i-1}, a_i) = r_i$ $(i = 1, \ldots, n)$
- $r_n \in F$

$L(M)$ : $M$ **が受理する語の全体** $\subset \Sigma^*$
$\cdots$ $M$ **が認識 (recognize) する言語**

$M$ **は言語** $L(M)$ **の "文法" で、**
$M$ **が受理する語は "文法に適っている"**

# 有限オートマトンによる語の受理

**有限オートマトン** $M = (Q, \Sigma, \delta, s, F)$ **が**
**語** $w = a_1 a_2 \cdots a_n$ **を受理 (accept) する**

$$\Updownarrow$$

$\exists r_0, r_1, \ldots, r_n \in Q$ :

- $r_0 = s$
- $\delta(r_{i-1}, a_i) = r_i$ $(i = 1, \ldots, n)$
- $r_n \in F$

$L(M)$ : $M$ **が受理する語の全体** $\subset \Sigma^*$
$\cdots$ $M$ **が認識 (recognize) する言語**

$M$ **は言語** $L(M)$ **の "文法" で、**
$M$ **が受理する語は "文法に適っている"**

# 演習問題

$\Sigma = \{a, b\}$ とする。

**次の言語を認識する有限オートマトンを構成し、状態遷移図で表せ**

**(1)** $A = \{a^{2n}b^{2m+1} \mid n, m \geq 0\}$
($a$ **が偶数個 (** $0$ **個も可) 続いた後に、**
$b$ **が奇数個続く)**

**(2)** $B = \{vabbaaaw \mid v, w \in \Sigma^*\}$
**(部分列として** $abbaa$ **を含む)**

# 演習問題

**ちょっとしたコツ (tips) :**

**「後に続く文字列が何だったら受理か」**

**が全く同じ状態は一つの状態にまとめられる。**

**これが違う状態はまとめられない。**
**(違う状態として用意する必要あり)**