

計算の理論

命令の実行 (= 「計算」) とは、

レジスタまたは主記憶の
現在の値 (状態) に従って、

その値を変更 (書込) すること

であった。

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いていたが、やはり本質的に違うもの。

- 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。

或る問題の「計算が可能」

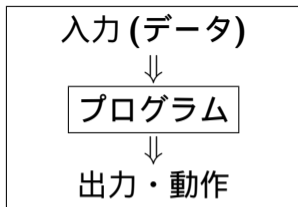


その計算を行なうプログラムが存在



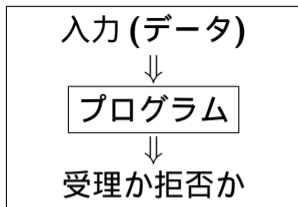
計算機の機能 (= 「計算」のモデル)
を決めて議論する

ここでは、代表的な「計算のモデル」を
幾つか紹介する。



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)



解くべき「問題」：入力を受理する条件

「問題」の例

入力の範囲: 文字 a, b から成る文字列

- a と b との個数が同じ
- a が幾つか続いた後に
 b が幾つか続いたもの
- a で始まり a, b が交互に並んで 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

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
「文法的に正しい」文字列だと思えば、

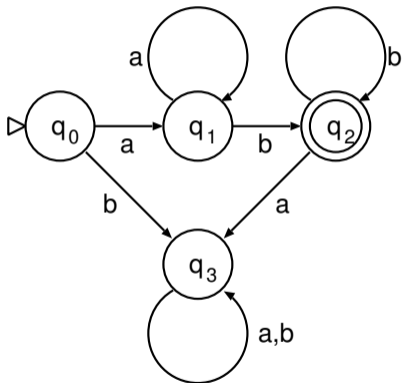
「問題」とは「文法 (言語)」である。

「文法的に正しい」かどうかの判定
… 「構文解析」

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形式的定義

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

ここに、

- Q : 有限集合 … 状態の集合
- Σ : 有限集合 … 入力文字の集合
“alphabet”
- $\delta : Q \times \Sigma \rightarrow Q$: 遷移関数
- $s \in Q$ … 初期状態
- $F \subset Q$ … 受理状態の集合

先の例では、

- $Q = \{q_0, q_1, q_2, q_3\}$

- $\Sigma = \{a, b\}$

- $\delta : Q \times \Sigma \rightarrow Q :$

	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

- $s = q_0 \in Q$

- $F = \{q_2\} \subset Q$

Σ : 入力文字の有限集合 \cdots **alphabet**

入力は Σ の元の有限列 (**語**)

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

その全体 Σ^*

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

言語 : Σ^* 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 “文法的に正しい”

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

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

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

$L(M)$: M が受理する語の全体
... M が認識する言語

M は言語 $L(M)$ の“文法”で、
 M が受理する語は“文法的に正しい”

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

ちょっとしたコツ (tips):

「後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)

有限オートマトンでの計算可能性問題

- 言語 $A \subset \Sigma^*$ に対し、
 A を認識する有限オートマトン M
 が存在するか？
- 有限オートマトンによって
 認識可能な言語はどのようなものか？

→ 正規言語・正規表現

語の演算

語 $v = a_1 \cdots a_k, w = b_1 \cdots b_l \in \Sigma^*$ に対し

$$vw := a_1 \cdots a_k b_1 \cdots b_l$$

: 連結・連接 (concatnation)

連接演算により Σ^* は単位的自由半群を成す

$S = (S, \cdot) : \text{半群}$



$\cdot : S \times S \longrightarrow S : \text{二項演算で結合律を満たす}$

語の演算

語 $v = a_1 \cdots a_k, w = b_1 \cdots b_l \in \Sigma^*$ に対し

$$vw := a_1 \cdots a_k b_1 \cdots b_l$$

: 連結・連接 (concatnation)

連接演算により Σ^* は単位的自由半群を成す

$$S = (S, \cdot) : \text{半群}$$



$\cdot : S \times S \longrightarrow S$: 二項演算で結合律を満たす

正規演算

言語 $A, B \subset \Sigma^*$ に対し、

- $A \cup B := \{w \mid w \in A \text{ または } w \in B\}$
: 和集合演算
- $AB = A \circ B := \{vw \mid v \in A, w \in B\}$
: 連結 (接続) 演算
- $A^* := \{w_1 w_2 \cdots w_n \mid n \geq 0, w_i \in A\}$
: star 演算

(言語全体の集合 $\mathcal{P}(\Sigma^*)$ 上の演算)

正規表現

- 各 **alphabet** $a \in \Sigma$ は正規表現
- 空列 ε は正規表現
- 空集合 \emptyset は正規表現
- 正規表現 R, S に対し
($R \cup S$) は正規表現 ($(R|S)$ とも書く)
- 正規表現 R, S に対し
($R \circ S$) は正規表現 ((RS) とも書く)
- 正規表現 R に対し R^* は正規表現
- 以上のものだけが正規表現

… 帰納的導出による定義

正規表現

- 各 **alphabet** $a \in \Sigma$ は正規表現
- 空列 ε は正規表現
- 空集合 \emptyset は正規表現
- 正規表現 R, S に対し
($R \cup S$) は正規表現 ($(R|S)$ とも書く)
- 正規表現 R, S に対し
($R \circ S$) は正規表現 ((RS) とも書く)
- 正規表現 R に対し R^* は正規表現
- 以上のものだけが正規表現

… 帰納的導出による定義

正規表現 R に対し、言語 $L(R)$ を次で定める。

- $L(a) = \{a\}$
- $L(\varepsilon) = \{\varepsilon\}$
- $L(\emptyset) = \emptyset$
- $L(R \cup S) = L(R) \cup L(S)$
- $L(R \circ S) = L(R) \circ L(S)$
- $L(R^*) = L(R)^*$

正規表現で表される言語 …… **正規言語**

定理:

L : 正規言語



L が或る有限オートマトンで認識される

このような一般論を考えるには、
有限オートマトンの概念を
少し一般化する方が良い。

… 非決定性有限オートマトン

定理:

L : 正規言語



L が或る有限オートマトンで認識される

このような一般論を考えるには、
有限オートマトンの概念を
少し一般化する方が良い。

… 非決定性有限オートマトン

非決定性 … あてずっぽう有り

状態の遷移先を一意に決めない
(幾つかあって分岐していく)

→ どれかが受理すれば OK!!

どの道を辿れば良いか知っていて、
受理が検証出来れば良い、

と考えることも出来る。

非決定性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 ... 状態の集合
- Σ : 有限集合 ... **alphabet**
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$: 遷移関数
... 可能な遷移先全体の集合
- $s \in Q$... 初期状態
- $F \subset Q$... 受理状態の集合

非決定性有限オートマトン

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

が、語 w を**受理**する



$$\exists a_1, a_2, \dots, a_n \in \Sigma \cup \{\varepsilon\} : w = a_1 a_2 \dots a_n$$

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

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

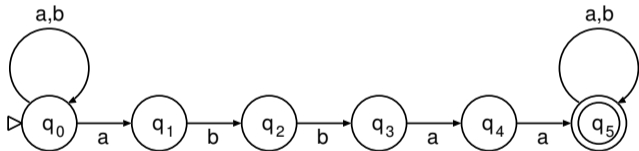
$L(M)$: M が受理する語の全体

… M が**認識**する言語

- $r_i \in \delta(r_{i-1}, \varepsilon)$ とは、
 「入力を読まずに
 状態 r_{i-1} から状態 r_i に移って良い」
 ということ
- $\delta(r_{i-1}, a_i) = \emptyset$ (矢印が出ていない)
 ということもある
 → 受理されない分岐の
 行き止まりに入ってしまった
 → 他の分岐が生きていれば問題無し

非決定性有限オートマトンの例

(状態遷移図による表示)



定理:

L が或る (決定性) 有限オートマトンで
認識される



L が或る非決定性有限オートマトンで
認識される

非決定性有限オートマトン M に対し、
 $L(M)$ を認識する決定性有限オートマトン M' が
構成できる

定理:

L が或る (決定性) 有限オートマトンで
認識される



L が或る非決定性有限オートマトンで
認識される

非決定性有限オートマトン M に対し、
 $L(M)$ を認識する決定性有限オートマトン M' が
構成できる

定理:

L : 正規言語



L が或る有限オートマトンで認識される

定理:

L : 正規言語



L が或る非決定性有限オートマトンで
認識される

正規言語:

- $L(a) = \{a\}$
- $L(\varepsilon) = \{\varepsilon\}$
- $L(\emptyset) = \emptyset$
- $L(R \cup S) = L(R) \cup L(S)$
- $L(R \circ S) = L(R) \circ L(S)$
- $L(R^*) = L(R)^*$

言語 A, B を認識する

非決定性有限オートマトンから、

言語 $A \cup B, A \circ B, A^*$ を認識する

非決定性有限オートマトンが
構成できれば良い。(それは比較的容易)

$A \cup B$ を認識する

非決定性有限オートマトンの構成

