

代表的な計算モデル

- 有限オートマトン
(有限状態機械, **finite automaton**)
- 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

語・言語

Σ : 入力文字の有限集合 …… **alphabet**

入力は Σ の元の有限列 (**語**, **word**)

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

その全体 Σ^*

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

言語 (language) : Σ^* の部分集合

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

… 言語 A に於いて “文法的に正しい”

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



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

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

$L(M) : M$ が受理する語の全体 $\subset \Sigma^*$
… 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)

演習問題

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

$S = (S, \cdot) : \text{半群 (semigroup)}$



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

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 (regular representation)

- 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

正規言語 (regular language)

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

「非決定性」とは … 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!!

- 手分けをして誰かが受理出来れば良い
- 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**, $\Sigma_\epsilon := \Sigma \cup \{\epsilon\}$
- $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$: 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による語の受理

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



$$\exists a_1, a_2, \dots, a_n \in \Sigma_\epsilon : w = a_1 a_2 \dots a_n$$

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

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

$L(M)$: M が受理する語の全体

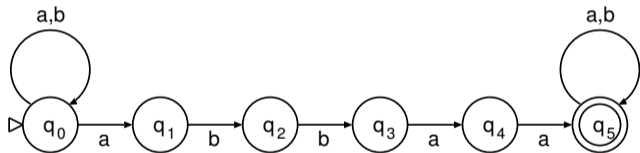
… M が認識する言語

非決定性有限オートマトンによる語の受理

- $r_i \in \delta(r_{i-1}, \varepsilon)$ 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 : 正規言語



L が或る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

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、

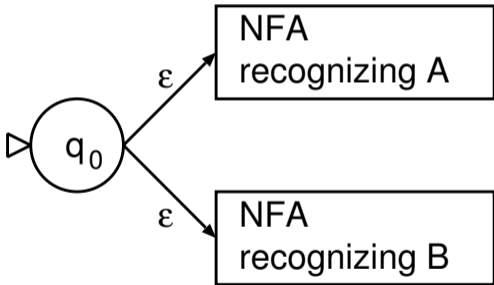
言語 $A \cup B, A \circ B, A^*$ を認識する

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



有限オートマトンでの計算可能性問題

非決定性有限オートマトンで認識できない
言語が存在する!!
(\iff 正規でない言語が存在する)

例: $A = \{a^n b^n \mid n \geq 0\}$
(a と b との個数が同じ)

証明は部屋割り論法
(の一種の pumping lemma)
による

Pumping Lemma(注入補題・反復補題)

正規言語 A に対し、

$\exists n \in \mathbf{N} :$

$\forall w \in A, |w| \geq n :$

$\exists x, y, z \in \Sigma^* : w = xyz$

(1) $y \neq \varepsilon$

(2) $|xy| \leq n$

(3) $\forall k \geq 0 : xy^kz \in A$

より強力な計算モデルが必要



- プッシュダウンオートマトン
- チューリングマシン

例: “文法的に正しい” 数式とは
どのようなものか?

簡単のため二項演算子のみ考えることにすれば、

- 単独の文字 (変数名) は式
- 式と式とを演算子で繋いだものは式
- 式を括弧で括ったものは式
- それだけ

→ これは式を作り出す規則とも考えられる

“文法的に正しい” 数式

初期記号 (開始変数) E から出発して、
次の規則のいずれかを

“非決定的に” 適用して得られるもの のみ

- $E \rightarrow A$
- $E \rightarrow EBE$
- $E \rightarrow (E)$
- $A \rightarrow$ 変数名のどれか
- $B \rightarrow$ 演算子のどれか
- 変数名・演算子・ (\cdot) は
それ以上書換えない (終端記号)

→ 生成規則 (書換規則)

生成規則・生成文法

生成規則を与えることでも

言語を定めることが出来る

→ 生成文法 (**generative grammar**)

生成規則による“正しい”語の生成

- 初期変数を書く
- 今ある文字列中の或る変数を生成規則のどれかで書換える
- 変数がなくなったら終わり

例: $\{a^{2n}b^{2m+1} \mid n, m \geq 0\}$

(a が偶数個 (0 個も可) 続いた後に、
 b が奇数個続く)

正規表現で表すと、 $(aa)^*b(bb)^*$

- $S \rightarrow aaS$
- $S \rightarrow bB$
- $B \rightarrow bbB$
- $B \rightarrow \varepsilon$

まとめて次のようにも書く

- $S \rightarrow aaS \mid bB$
- $B \rightarrow bbB \mid \varepsilon$

生成規則・生成文法

実際の (自然言語を含めた) “文法” では、
或る特定の状況で現われた場合だけ
適用できる規則もあるだろう

そのような生成規則は例えば次の形:

- $uAv \rightarrow uvw$

$u, v \in \Sigma^*$: 文脈 (context)

変数 A が uAv の形で現われたら、
語 $w \in \Sigma^*$ で書換えることができる

生成文法の形式的定義

- V : 有限集合 (変数の集合)
- Σ : 有限集合 (終端記号の集合)
ここに $V \cap \Sigma = \emptyset$
- R : 有限集合 $\subset (V \cup \Sigma)^* \times (V \cup \Sigma)^*$
(規則の集合)
- $S \in V$: 開始変数

$(v, w) \in R$ が生成規則 $v \rightarrow w$ を表す

文脈自由文法 (context-free grammar)

文脈が全て空列 ε

即ち、規則が全て $A \rightarrow w$ ($A \in V$) の形

文脈自由文法の形式的定義

- V : 有限集合 (変数の集合)
- Σ : 有限集合 (終端記号の集合)
ここに $V \cap \Sigma = \emptyset$
- R : 有限集合 $\subset V \times (V \cup \Sigma)^*$
(規則の集合)
- $S \in V$: 開始変数

$(A, w) \in R$ が生成規則 $A \rightarrow w$ を表す

- 例: 言語 $A = \{a^n b^n \mid n \geq 0\}$ は
正規言語ではないが文脈自由言語である:
- $S \rightarrow aSb \mid \varepsilon$

従って、

文脈自由言語は正規言語より真に広い!!

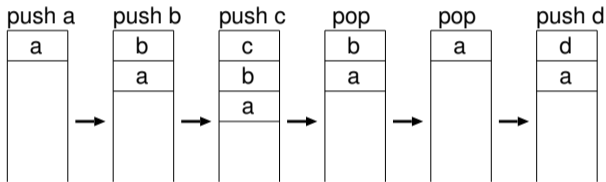
さて、正規言語を計算するモデルが
有限オートマトンであった

文脈自由言語を計算するモデル
... プッシュダウンオートマトン

プッシュダウンオートマトン

(非決定性) 有限オートマトンに

プッシュダウンスタックを取り付けたもの



無限 (非有界) の情報を保持できるが、
読み書きは先頭だけ

... LIFO (Last In First Out)