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？

定理：

L：正規言語

$\Updownarrow$

L が或る有限オートマトンで認識される

有限オートマトンでの計算可能性問題

定理：

L：正規言語

$\Updownarrow$

L が或る有限オートマトンで認識される

実は、
有限オートマトンで認識できない言語が存在する !!
（$\Longleftrightarrow$ 正規でない言語が存在する）

有限オートマトンでの計算可能性問題

定理：

L：正規言語

$\Updownarrow$

L が或る有限オートマトンで認識される

実は、
有限オートマトンで認識できない言語が存在する !!
（$\Longleftrightarrow$ 正規でない言語が存在する）

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？

$\longrightarrow$

言語 A が
正規言語である（FA で認識される）かどうか、
の良い判定基準は？

集合・写像などの言葉を用いて記述する練習

演習問題2:Σ を alphabet とする。以下を記述せよ。

(3) L：言語

(a) 文字 $a \in \Sigma$ に対し、語に左（resp. 右）から文字 a を連接させる写像 ℓ_a (resp. r_a)

(b) 語 $w \in \Sigma^*$ に対し、語に左（resp. 右）から文字列 w を連接させる写像 ℓ_w (resp. r_w)（語の長さ $|w|$ に関する帰納的定義で）

(c) 語 $w \in \Sigma^*$ の後に連接すると L の元になる語全体の成す集合 $S_{\mathrm{L}}(w)$ を与える写像 S_{L}

(4) $M = (Q, \Sigma, \delta, s, F)$**：有限オートマトン**

(a) 状態 $q \in Q$ **にいる所から出発して語** $w \in \Sigma^*$ **を読んだ後の状態** $\widetilde{\delta}(q, w)$ **を与える写像** $\widetilde{\delta}$ **（語の長さ** $|w|$ **に関する帰納的定義で）**

(b) 特に、M **が語** $w \in \Sigma^*$ **を読んだ後の状態を与える写像** $\widetilde{\delta}_0$

(c) M **が認識する言語** $L(M)$

(d) 状態 $q \in Q$ **にいる所から出発して、その後に読めば受理される語全体の成す集合** $\varphi_M(q)$ **を与える写像** φ_M

有限オートマトンでの計算可能性問題

有限オートマトンで認識できる

$\Longleftrightarrow$ **“待ち” が有限種類**

$\ell_w : \Sigma^* \longrightarrow \Sigma^*$ **：“左平行移動”**

$$v \longmapsto wv$$

言語 $\mathrm{L} \in \mathcal{P}(\Sigma^*)$ **に対し、**

$S_{\mathrm{L}} : \Sigma^* \longrightarrow \mathcal{P}(\Sigma^*)$ **：“待ち” の集合**

$$w \longmapsto \{v \in \Sigma^* \,|\, wv \in \mathrm{L}\} = \ell_w^{-1}(\mathrm{L})$$

$$\#\mathrm{Im}S_{\mathrm{L}} < \infty \Longleftrightarrow \exists M : \mathrm{L} = \mathrm{L}(M)$$

有限オートマトンでの計算可能性問題

有限オートマトンで認識できる

$\Longleftrightarrow$ **“待ち” が有限種類**

$\ell_w : \Sigma^* \longrightarrow \Sigma^*$ **：“左平行移動”**

$$v \longmapsto wv$$

言語 $L \in \mathcal{P}(\Sigma^*)$ **に対し、**

$S_L : \Sigma^* \longrightarrow \mathcal{P}(\Sigma^*)$ **：“待ち” の集合**

$$w \longmapsto \{v \in \Sigma^* \,|\, wv \in L\} = \ell_w^{-1}(L)$$

$$\#\mathrm{Im}S_L < \infty \Longleftrightarrow \exists M : L = L(M)$$

有限オートマトンでの計算可能性問題

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

例：$A = \{a^n b^n | n \geq 0\}$ （a と b との個数が同じ）
実際 $w_n = a^n b$ に対する $S_L(w_n)$ が全て異なる

一般には、証明には部屋割り論法
（の一種の pumping lemma）
を利用することが多い

有限オートマトンでの計算可能性問題

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

例：$A = \{a^n b^n | n \geq 0\}$ （a と b との個数が同じ）
実際 $w_n = a^n b$ に対する $S_L(w_n)$ が全て異なる

一般には、証明には部屋割り論法
（の一種の pumping lemma）
を利用することが多い

有限オートマトンでの計算可能性問題

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

例：$A = \{a^n b^n | n \geq 0\}$ （a と b との個数が同じ）
実際 $w_n = a^n b$ に対する $S_L(w_n)$ が全て異なる

一般には、証明には部屋割り論法
（の一種の pumping lemma）
を利用することが多い

有限オートマトンでの計算可能性問題

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

例： $A = \{a^n b^n | n \geq 0\}$ **（a と b との個数が同じ）**
実際 $w_n = a^n b$ **に対する** $S_L(w_n)$ **が全て異なる**

一般には、証明には部屋割り論法
（の一種の pumping lemma）
を利用することが多い

Pumping Lemma（注入補題・反復補題）

正規言語 $A \subset \Sigma^*$ に対し、

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

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

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

(1) $y \neq \varepsilon$
(2) $|xy| \leq n$
(3) $\forall k \geq 0 : xy^k z \in A$

有限オートマトンで認識できる／ない言語の例

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

- 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, b が交互に並んで、a で始まり 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

このうちで、

有限オートマトンで認識できる言語は？

有限オートマトンで認識できない言語が存在する

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

有限オートマトンで認識できない言語が存在する

⇓

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**

有限オートマトンで認識できない言語が存在する

⇓

有限オートマトンより強力な計算モデル

⇓

正規言語より広い範囲の言語を扱う

⇓

生成規則による言語の記述（生成文法）

有限オートマトンで認識できない言語が存在する

⇓

有限オートマトンより強力な計算モデル

⇓

正規言語より広い範囲の言語を扱う

⇓

生成規則による言語の記述（生成文法）

例：“文法に適っている” 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

$\longrightarrow$ これは式を作り出す規則とも考えられる

例：“文法に適っている” 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

$\longrightarrow$ これは式を作り出す規則とも考えられる

例：“文法に適っている” 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**

$\longrightarrow$ これは式を作り出す規則とも考えられる

“文法に適っている” 数式

初期記号（開始変数）E から出発して、
次の規則のいづれかを
“非決定的に” 適用して得られるもののみ

- $\mathrm{E} \to \mathrm{A}$
- $\mathrm{E} \to \mathrm{EBE}$
- $\mathrm{E} \to (\mathrm{E})$
- $\mathrm{A} \to$ **変数名のどれか**
- $\mathrm{B} \to$ **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）

$\longrightarrow$ **生成規則（書換規則）**

“文法に適っている” 数式

初期記号（開始変数）E から出発して、
次の規則のいづれかを
“非決定的に” 適用して得られるもののみ

- $\mathrm{E} \to \mathrm{A}$
- $\mathrm{E} \to \mathrm{EBE}$
- $\mathrm{E} \to (\mathrm{E})$
- $\mathrm{A} \to$ **変数名のどれか**
- $\mathrm{B} \to$ **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）

$\longrightarrow$ **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

$\longrightarrow$ **生成文法 (generative grammar)**

生成規則による “文法に適っている” 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**

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

$\longrightarrow$ **生成文法 (generative grammar)**

生成規則による “文法に適っている” 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