

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

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)

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

\iff “待ち” が有限種類

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

$$v \longmapsto wv$$

言語 $L \in \mathcal{P}(\Sigma^*)$ に対し、

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

$$w \longmapsto \{v \in \Sigma^* \mid mv \in L\} = \ell_w^{-1}(L)$$

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

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)
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

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)

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

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)

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

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



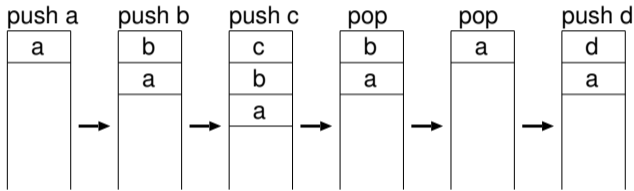
より強力な計算モデルが必要



- プッシュダウンオートマトン
- チューリングマシン

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)**

有限オートマトンより強力な計算モデルが必要



正規言語より広い範囲の言語を考えることが必要



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

→ 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

→ 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

→ 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)

“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“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