

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

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 : 文脈自由言語



L が或るプッシュダウンオートマトンで
認識される

本質的な違いは？

文脈自由言語は再帰 (recursion) を記述できる

正規言語・文脈自由言語と再帰

- 正規言語は繰返しを記述できる
- 文脈自由言語は再帰を記述できる
- 再帰の実装にはスタックを要す
- 文脈自由言語の生成規則は次の形に出来る
 - ★ $X \longrightarrow YZ$ ($X, Y, Z \in V$)
 - ★ $X \longrightarrow x$ ($X \in V, x \in \Sigma_\varepsilon$)(**Chomsky** の標準形)
- 正規言語の生成規則は次の形に出来る
 - ★ $X \longrightarrow xY$ ($X, Y \in V, x \in \Sigma$)
 - ★ $X \longrightarrow x$ ($X \in V, x \in \Sigma_\varepsilon$)特に、末尾再帰であり再帰の除去可能

文脈自由言語の Pumping Lemma

文脈自由言語 A に対し、

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

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

$\exists u, v, x, y, z \in \Sigma^* : w = uvxyz$

(1) $vy \neq \varepsilon$ (即ち $v \neq \varepsilon$ または $y \neq \varepsilon$)

(2) $|vxy| \leq n$

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

プッシュダウンオートマトンでは

認識できない言語の例

同じ文字列 2 回の繰返しから成る文字列全体

$$A = \{ww \mid w \in \Sigma^*\}$$

入力を読み直せないのが弱点

→ より強力な計算モデルが必要

一つの方法としては、

入力を覚えておくために

プッシュダウンスタックをもう一つ

使えることにする

実際これで真により強い計算モデルが得られる

しかし、通常はこれと同等な

次のような計算モデルを考える

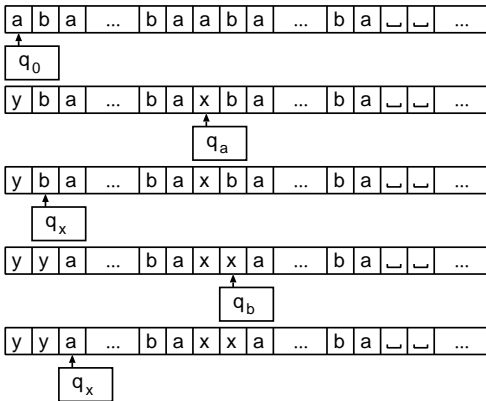
… チューリングマシン

チューリングマシン

- 有限個の内部状態を持つ
- 入力データはテープ上に一区画一文字ずつ書き込まれて与えられる
- データを読み書きするヘッドがテープ上を動く
- 遷移関数は次の形：
内部状態とヘッドが今いる場所の文字とによって、その場所の文字を書き換え、次の内部状態に移り、ヘッドを左か右かに動かす
- 受理状態または拒否状態に達したら停止するが、停止しないこともある

(非決定性) チューリングマシンによる

言語 $A = \{ww \mid w \in \Sigma^*\}$ の認識



チューリングマシンによる言語の認識

チューリングマシン T が言語 A を認識する



$$A = \left\{ w \in \Sigma^* \mid \begin{array}{l} \text{入力 } w \text{ に対し、} \\ \text{受理状態で停止する} \\ \text{遷移が存在} \end{array} \right\}$$



$w \in A \iff$ 入力 w に対し、
受理状態で停止する遷移が存在

チューリングマシンによる言語の判定

チューリングマシン T が言語 A を判定する



T は A を認識し、
かつ、全ての入力に対し必ず停止する



$w \in A \iff$ 入力 w に対し、
受理状態で停止する遷移が存在
かつ

$w \notin A \iff$ 入力 w に対し、
すべての遷移が拒否状態で停止

Church-Turing の提唱

「全てのアルゴリズム（計算手順）は、
チューリングマシンで実装できる」

（アルゴリズムと呼べるのは
チューリングマシンで実装できるものだけ）

… 「アルゴリズム」の定式化

何故「チューリングマシン」なのか？

- およそ計算機で実行したいことは模倣可能
(非有界のメモリにランダムアクセスできる
計算機モデル)
- 多少モデルを変更しても強さが同じ
(モデルの頑強性)
 - ★ テープが両方に無限に伸びているか
 - ★ ヘッドが動かないことがあっても良いか
 - ★ 複数テープチューリングマシン
 - ★ 決定性 / 非決定性 などなど

プログラム内蔵方式 (von Neumann 型)

… プログラムもデータとして保持

→ 一つの機械で様々な計算を柔軟に実現

同様の働きをするチューリングマシンが構成できる

… 普遍チューリングマシン

(万能チューリングマシン)

(universal Turing machine)

普遍チューリングマシン

全てのチューリングマシンの動作を模倣する

- 入力： $(\langle M \rangle, w)$
 - ★ $\langle M \rangle$ ：機械 M の符号化（プログラムに相当）
 - ★ w ： M に与える入力データ

- 出力：機械 M が入力 w を受理するかどうか

普遍チューリングマシン

普遍チューリングマシンとは、
言語

$$A_{\text{TM}} = \left\{ (\langle M \rangle, w) \mid \begin{array}{l} \langle M \rangle : \text{TM } M \text{ の符号化} \\ M \text{ が入力 } w \text{ を受理} \end{array} \right\}$$

を認識するチューリングマシン

普遍チューリングマシンが存在

$\iff A_{\text{TM}}$ がチューリングマシンで認識可能

定理

言語

$$A_{\text{TM}} = \left\{ (\langle M \rangle, w) \mid \begin{array}{l} \langle M \rangle : \text{TM } M \text{ の符号化} \\ M \text{ が入力 } w \text{ を受理} \end{array} \right\}$$

は認識可能だが、判定可能ではない。

証明は一種の対角線論法による
(Russell のパラドックス風)

対角線論法の例：Russell のパラドックス

$X := \{A \mid A \notin A\}$ とせよ

$X \in X$ であるか？

- $X \in X$ と仮定すると、定義より $X \notin X$
- $X \notin X$ と仮定すると、定義より $X \in X$

→ どちらにしても**矛盾** !!

A_{TM} の判定不可能性

A_{TM} を判定する TM U があったとする。

入力 $\langle M \rangle$ に対し、

- M が $\langle M \rangle$ を受理するなら拒否
- M が $\langle M \rangle$ を拒否するなら受理

となる TM D が (U を使って) 作れる。

これに、入力 $\langle D \rangle$ を喰わせよ。