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** $\cdots$ **状態の集合**
- Σ : **有限集合** $\cdots$ **入力文字の集合: "alphabet"**
- $\delta : Q \times \Sigma \to Q$ ： **遷移関数**
- $s \in Q$ $\cdots$ **初期状態**
- $F \subset Q$ $\cdots$ **受理状態の集合**

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 $w = a_1 a_2 \cdots a_n \in \Sigma^*$ **を受理 (accept) する**

$$\Updownarrow$$

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

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

$L(M)$ **:** M **が受理する語の全体** $\subset \Sigma^*$
$\cdots$ 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 "文法に適っている"**

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？

- 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？

$\longrightarrow$ **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**

$$vw := a_1 \cdots a_k b_1 \cdots b_l \in \Sigma^*$$

: 連結・連接 (concatnation)

連接演算により Σ^* **は単位的自由半群を成す**

$S = (S, \cdot)$ **: 半群 (semigroup)**

$$\Updownarrow$$

$\cdot : S \times S \longrightarrow S$ **: 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、

- $A \cup B := \{w | w \in A$ または $w \in B\}$

  ： 和集合演算
- $AB = A \circ B := \{vw | v \in A, w \in B\}$

  ： 連結 (連接) 演算
- $A^* := \{w_1 w_2 \cdots w_n | n \geq 0, w_i \in A\}$

  ： star 演算

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

正規表現 (regular representation)

- 空集合記号 $\emptyset$ は正規表現
- 空列記号 ε は正規表現
- 各 alphabet $a \in \Sigma$ 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

$\cdots$ 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：

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

正規表現で表される言語 $\cdots$ **正規言語**

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** $\cdots$ **状態の集合**
- Σ **: 有限集合** $\cdots$ **alphabet,** $\quad \Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$**: 遷移関数**
  $\cdots$ **可能な遷移先全体の集合を与える**
- $s \in Q$ $\cdots$ **初期状態**
- $F \subset Q$ $\cdots$ **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 $w \in \Sigma^*$ を受理する

$$\Updownarrow$$

$\exists a_1, a_2, \cdots, a_n \in \Sigma_\varepsilon : w = a_1 a_2 \cdots a_n$

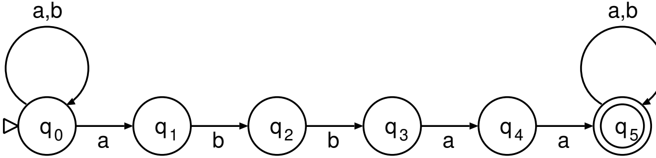
$\exists r_0, r_1, \ldots, r_n \in Q :$

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

$L(M)$ **：** M **が受理する語の全体**

$\cdots$ M **が認識する言語**

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： **正規言語**

$\Updownarrow$

Ｌ **が或る決定性有限オートマトンで認識される**

$\Updownarrow$

Ｌ **が或る非決定性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

- 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？

$\longrightarrow$ 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!!
($\Longleftrightarrow$ 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

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!!
($\Longleftrightarrow$ 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)**

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

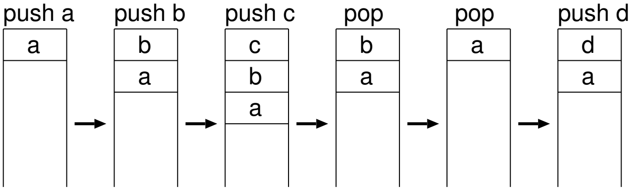
より強力な計算モデルが必要

$\Downarrow$

- 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

$\cdots$ **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

$\Downarrow$

正規言語より広い範囲の言語を考えることが必要

$\Downarrow$

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**

“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

- $\mathsf{E} \to \mathsf{A}$
- $\mathsf{E} \to \mathsf{EBE}$
- $\mathsf{E} \to (\mathsf{E})$
- $\mathsf{A} \to$ **変数名のどれか**
- $\mathsf{B} \to$ **演算子のどれか**
- **変数名・演算子・(・) 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

⟶ **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

例: $\{a^{2n}b^{2m+1} | n, m \geq 0\}$

(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

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

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:

- $u\textcolor{blue}{A}v \to u\textcolor{blue}{w}v$

$u, v \in \Sigma^*$: **文脈 (context)**

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

生成文法の形式的定義

$$G = (V, \Sigma, R, S)$$

- 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** $v \to w$ **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 $A \to w$ ($A \in V$) 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 $A \to w$ を表す

例: 言語 $A = \{a^n b^n | n \geq 0\}$ **は**
正規言語ではないが文脈自由言語である:

- $S \to aSb | \varepsilon$

従って、

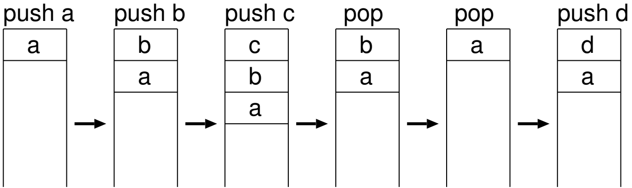
$\boxed{\text{文脈自由言語は正規言語より真に広い!!}}$

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
$\cdots$ **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

$\cdots$ **LIFO (Last In First Out)**

プッシュダウンオートマトンの形式的定義

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

- Q : **有限集合** $\cdots$ **状態の集合**
- Σ : **有限集合** $\cdots$ **alphabet**
- Γ : **有限集合** $\cdots$ **stack alphabet**
  $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$, $\Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$ **と置く**
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$
  : 遷移関数 $\cdots$ **可能な遷移先全体**
- $s \in Q$ $\cdots$ **初期状態**
- $F \subset Q$ $\cdots$ **受理状態の集合**

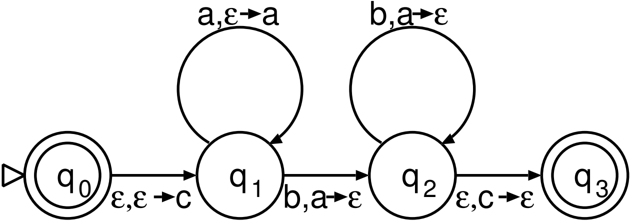
$$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$$

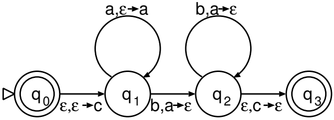
- $(r, y) \in \delta(q, a, x)$ **とは、**
  「入力 a を読んだとき、
  状態 q でスタックの先頭が x なら、
  スタックの先頭を y に書換えて、
  状態 r に移って良い」
  ということ (`pop; push y`**)**

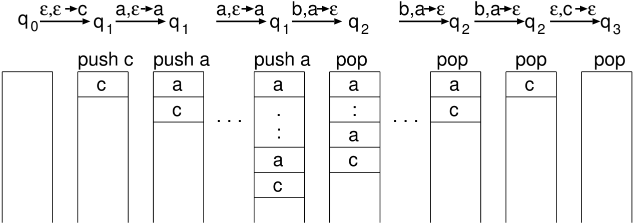
- $x = y$ **は書き換え無し**
- $x = \varepsilon$ **は push のみ**
- $y = \varepsilon$ **は pop のみ**
- $a = \varepsilon$ **は入力を読まずに遷移**

例: 言語 $A = \{a^n b^n | n \geq 0\}$ を認識する プッシュダウンオートマトン

$$\Sigma = \{a, b\}, \qquad \Gamma = \{a, b, c\}$$



PDA  による 文字列 $a^n b^n$ の受理



定理:

$$\mathrm{L}:\ \text{文脈自由言語}$$

$$\Updownarrow$$

L が或るプッシュダウンオートマトンで認識される

文脈自由言語の例

回文全体の成す言語は文脈自由

- $S \to aSa|bSb|a|b|\varepsilon$

問: 回文全体の成す言語を認識する
プッシュダウンオートマトンを構成せよ