オートマトンを
正規表現に変換する
アルゴリズムの話

def：言語の演算

言語 L_1, L_2 に対して

- $L_1 + L_2 = L_1 \cup L_2$
- $L_1 \cdot L_2 = \{ w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$
- $L_1^* = \{\varepsilon\} \cup L_1 \cup (L_1 \cdot L_1) \cup (L_1 \cdot L_1 \cdot L_1) \cup \cdots$ とする.

def: 正規表現として以下を考える.

- ϕ (空集合を表す)
- ε ($\{\varepsilon\}$を表す)
- a ($a \in \Sigma$, $\{a\}$を表す)
- $r_1 + r_2$ ($L(r_1) + L(r_2)$を表す)
- $r_1 r_2$ ($L(r_1) \cdot L(r_2)$を表す)
- r^* ($L(r)^*$を表す)

クイズ：アルファベット $\Sigma = \{a, b\}$ 上で

- a がちょうど偶数回現れる文字列を受理する正規表現は？
- a がちょうど3の倍数回現れる文字列を受理する正規表現は？

クイズ: アルファベット$\Sigma = \{a, b\}$上で

- aがちょうど偶数回現われる文字列を受理する正規表現は?

  $(ab^*a + b)^*$

- aがちょうど3の倍数回現われる文字列を受理する正規表現は?

  $(ab^*ab^*a + b)^*$

クイズ: アルファベット $\Sigma = \{a, b\}$ 上で

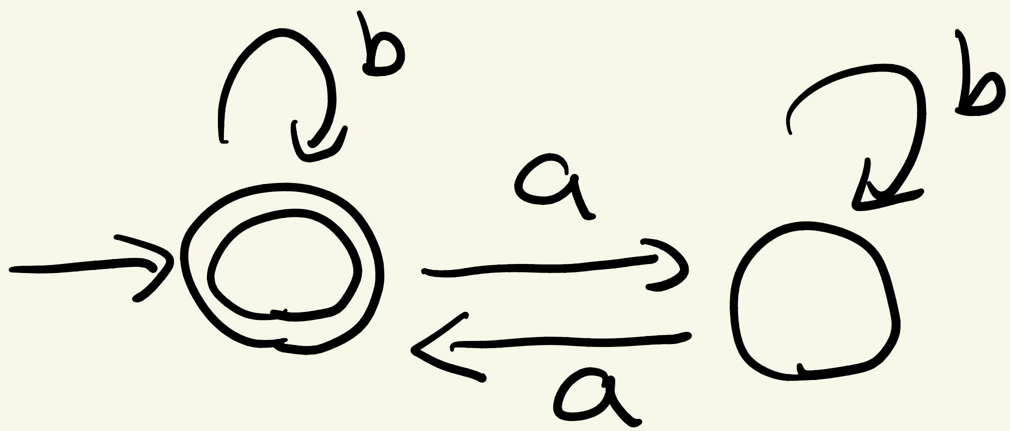
- a がちょうど偶数回現れる文字列を受理する ~~正規表現~~ は?

  オートマトン

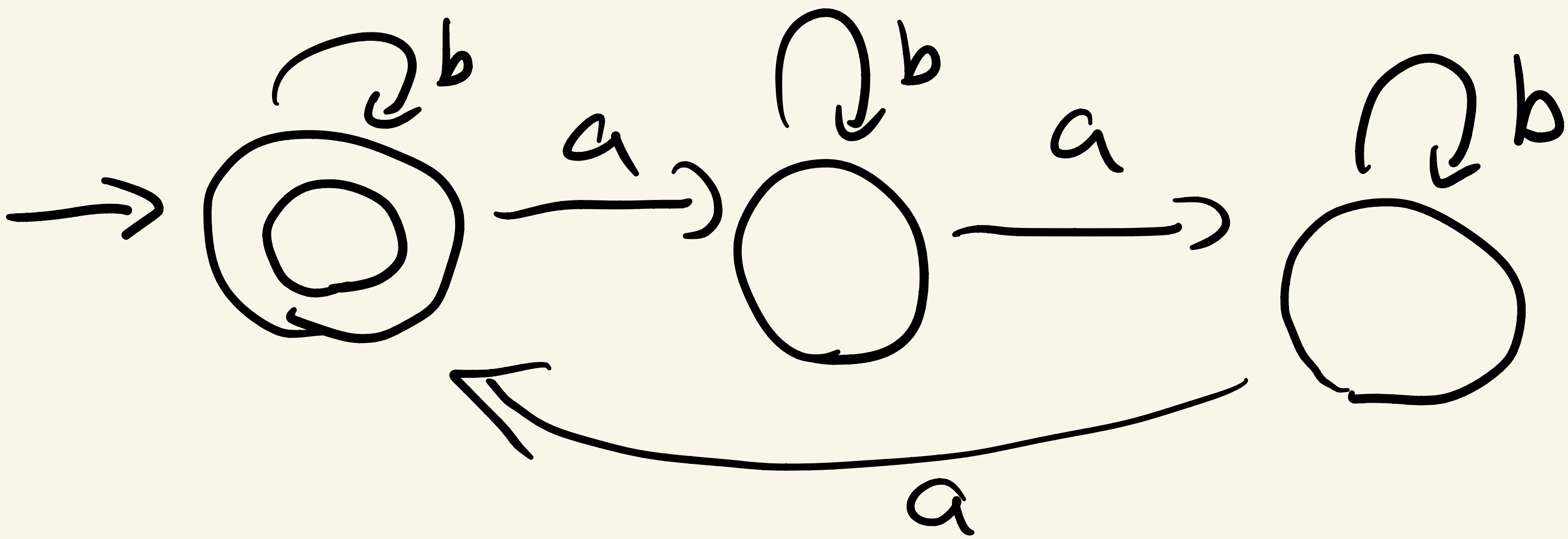
- a がちょうど3の倍数回現れる文字列を受理する ~~正規表現~~ は?

  オートマトン

オートマトンは簡単：

- 

（aが偶数回現れる文字列を受理）

- 

（aが3の倍数回現れる文字列を受理）

オートマトンはつくれたので
あとはこれを正規表現に変換
できればよい!

これに言語の等式

$$X = AX + B$$

の解法をつかう。

thm: 言語 A, B に対し

$X = AX + B$ をみたす言語 X は

$X = A^* B$ で与えられる.

例: • $X = a \cdot X + \varepsilon$ をみたす

言語 X は正規表現 a^* で与えられる。

• $X = aX + b$ をみたす

言語 X は正規表現 $a^* b$ で与えられる。

アルゴリズム: 与えられたDFAに対し

- 状態数だけ変数を用意:

  $$X_k = \{ s \in \Sigma^* \mid \delta^*(k, s) \in F \}$$

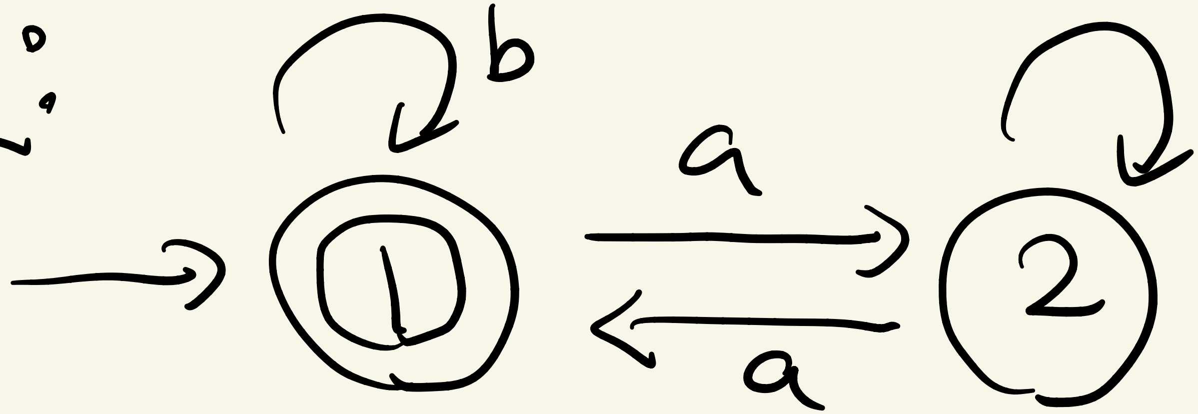
- 各変数に対して

  $$X_k = A_1 X_1 + \cdots + A_n X_n + \boxed{}$$

  の形の式をたてる.

- 始状態に対応する変数の解を出力

例：



の方程式は

$$\begin{cases} X_1 = bX_1 + aX_2 + \boxed{\varepsilon} \\ X_2 = aX_1 + bX_2 + \boxed{\emptyset} \end{cases}$$

。

X_2について解く：

$X_2 = bX_2 + aX_1$ より

$$X_2 = b^*(aX_1) = b^*aX_1 .$$

$X_1 = bX_1 + aX_2 + \varepsilon$ から

$X_2 = b^*aX_1$ をつかって X_2 を消去.

$$X_1 = bX_1 + ab^*aX_1 + \varepsilon$$

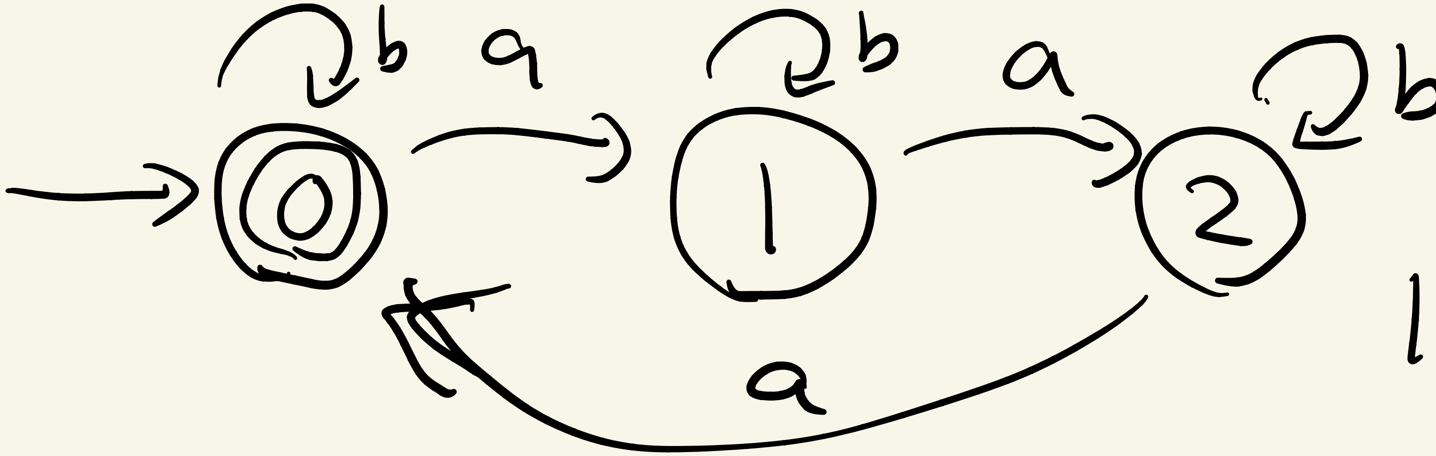
$$= (ab^*a + b)X_1 + \varepsilon.$$

これを解くと

$$X_1 = (ab^*a + b)^*\varepsilon$$

$$= \underline{(ab^*a + b)^*}.$$

例：



は

$$\begin{cases} X_0 = bX_0 + aX_1 + \varepsilon \\ X_1 = bX_1 + aX_2 + \emptyset \\ X_2 = bX_2 + aX_0 + \emptyset. \end{cases}$$

これを解くと

$$X_0 = (ab^*ab^*a+b)^*.$$

まとめ

既知の言語A, Bに対し

$$X = AX + B$$

をみたすような言語XはA^*B。

これを応用してオートマトンを
正規表現に変換するアルゴリズムが作れる。

(参考: Lawson の Finite Automata)

https://gist.github.com/saitouena/66955533fff8efc64612f2ce39562d9f