# 自然言語処理論 I

2.オートマトンと言語



## 言語の形式的モデル

- 形式言語 (formal language)
  - 記号列の部分集合
- 言語の形式的モデル
  - 形式言語の定義を与える
    - どのような記号列が言語で，どのような記号列が言語でないかを形式的に定義
  - 自然言語も記号列の集合
    - ○ 私は今日学校へ行った
    - × 行った今日へ学校は私
  - 計算機が自然言語を理解するための枠組のひとつ



## 言語の形式的モデル

- 有限オートマトン
- 正規言語、正規表現
- 形式文法
  - 文脈自由文法

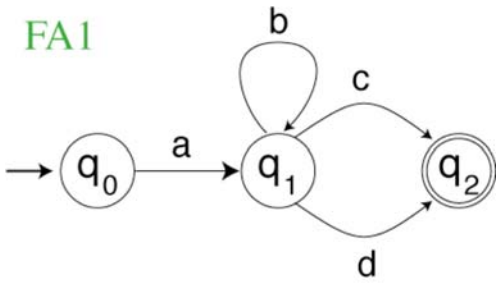


## 有限オートマトン

- Finite Automaton (FA)
- 入力は文字列
- 入力文字列を受理するかしないかを判定
- 状態遷移図で表現される

# FAの例

- aで始まり,bを何回か繰り返し,最後にcまたはdで終わる文字列を受理するFA




- ○ abc
- ○ abbbd
- × acbbc
- × abb
- × xyz



# FAの定義

- $M =< K, S, P, q_0, F >$
  - 状態集合 $K = \{q_0, q_1, ..., q_n\}$
  - 入力文字集合 $S$
  - 状態遷移関数の集合 $P = \{\delta(q_i, a) = q_j, ...\}$
  - 初期状態 $q_0$
  - 最終状態 $F$
- FA1のオートマトンの定義は?



# 正規言語

- 正規言語 (regular language)
  - FAが受理する言語
  - 正規言語は正規表現で表すことが可能

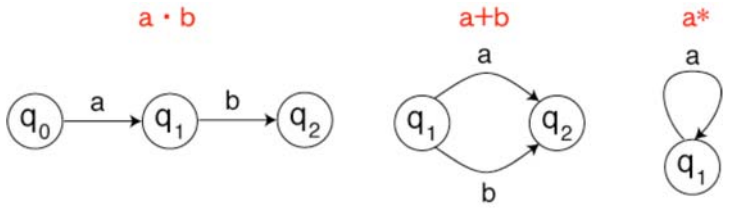


# 正規表現

- 正規表現 (regular expression)
  - 3つの演算子の組み合わせによって表現された文字列の集合
  - * (閉包): ある記号の0個以上の列
    - ✦ a*
  - ・(連結): 2つの記号を並べたもの
    - ✦ a ・ b　または　a b
  - +(和): 2つの記号のどちらでもよい
    - ✦ (a + b)・c
- FA1が受理する言語の正規表現は?

# FAと正規表現

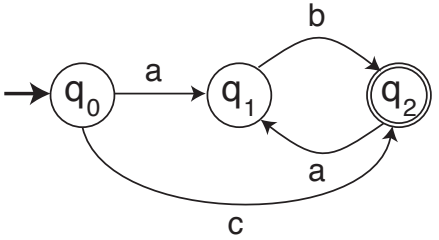
- FAで受理される文字列の集合は正規表現で表現できる
- 正規表現が表す文字列の集合を受理するFAは必ず存在する
- 演算子と状態遷移図の対応






# FAから正規表現へ

- FAの例

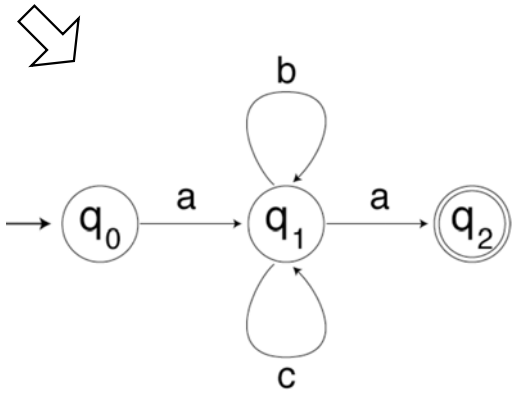



(ab + c) (ab)*



# 正規表現からFAへ

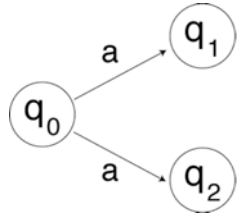
- a(b+c)*a






# 非決定性有限オートマトン

- 有限オートマトンが決定的であるとは?
  - 遷移先が常に一意に決まる
- 非決定性有限オートマトン
  - nondeterministic finite automaton
  - 決定的ではない有限オートマトン



# 非決定性有限オートマトン

- 受理の判定
  - 1つの入力記号列に対し、状態遷移は複数ある
  - どれかひとつでも最終状態に到達すれば受理
- 定義
  - $M =< K, S, P, q_0, F >$
  - 状態遷移関数P　$\delta(q_i, a) = \{q_{i1}, ..., q_{in}\}$
- 決定性FAと非決定性FAは等価
  - 非決定性FAと同じ言語を受理する決定性FAは必ず存在する



# 非決定性FAの例

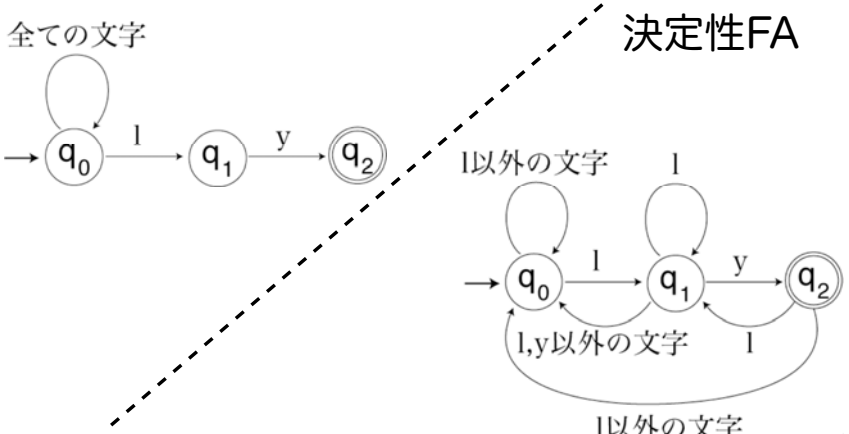
- lyで終わる英単語を受理するオートマトン
  - 非決定性FAの場合 (→次ページ)
  - 決定性FAの場合 (→次ページ)
- 非決定性にするとオートマトンが単純になる



# 英単語(-ly)を受理するFSA

非決定性FA

決定性FA









# FAの限界

- FAで受理できない言語がある
  - 例: $\{a^n b^n | n = 0, 1, 2, ...\}$
  - 状態数が有限であるFAでは実現不可能
- FAで自然言語を受理できるか?
  - 答えはNo
  - 例: (that)$_i$ he is young (is false)$_i$

that ... that that he is young is false is false ... is false

# 形式文法(formal grammar)

- 形式文法とは?
  - 書き換え規則の集合
  - 規則を順次適用することにより言語を生成する
- 例

| | |
|---|---|
| 文→主語 述語 | 名詞→太郎 |
| 主語→名詞 助詞 | 助詞→は \| が |
| 述語→動詞句 | 動詞→走り \| 走っ |
| 動詞句→動詞 | 助動詞→た \| だ \| そう |
| 動詞句→動詞句 助動詞 | |



# 形式文法の定義

- $G=<V_N, V_T, P, \sigma>$
  - $V_N$: 非終端記号(nonterminal symbol)の集合
  - $V_T$: 終端記号(terminal symbol)の集合
  - P: 書き換え規則(rewriting rule)の集合
  - $\sigma$ : 初期記号(initial symbol)または開始記号(start symbol)



# 記号列の導出

- 導出
  - 規則を使って記号列を生成すること
- 例:

文⇒主語 述語
⇒名詞 助詞 述語
⇒名詞 助詞 動詞句
⇒名詞 助詞 動詞 助動詞
⇒太郎 助詞 動詞 助動詞
⇒太郎 が 動詞 助動詞
⇒太郎 が 走っ 助動詞
⇒太郎 が 走っ た



# 形式文法と言語

- 言語L(G)の定義
  - 初期記号から，文法Gの書き換え規則によって生成される終端記号列の集合
  - $L(G) = \{w | w \in V_T^*, \sigma \overset{*}{\Rightarrow} w\}$
    - ✦ $\sigma \overset{*}{\Rightarrow} w$ は、初期記号から書き換え規則を0回以上適用してwが生成されることを表す

# 文法のクラス

- 4種類の文法
  - 0型文法
    - ✦ $\alpha \rightarrow \beta$
    - ✦ $\alpha, \beta$ は任意の$V=V_N \cup V_T$の列、$\alpha$ は$V_N$を1つ以上含む
  - 1型文法(文脈依存文法, context sensitive grammar)
    - ✦ $\alpha \rightarrow \beta$　　( $|\alpha| \leqq |\beta|$ )
  - 2型文法(文脈自由文法, context free grammar)
    - ✦ $A \rightarrow \beta$　　( $A \in V_N$ )
  - 3型文法(正規文法, regular grammar)
    - ✦ $A \rightarrow a\ B$　または　$A \rightarrow a$　　($A,B \in V_N$ , $a \in V_T$ )



# 文法,言語,オートマトン

| 文法 | 言語 | オートマトン |
|---|---|---|
| 正規文法<br>(3型文法) | 正規言語<br>(3型言語) | 有限オートマトン |
| 文脈自由文法<br>(2型文法) | 文脈自由言語<br>(2型言語) | プッシュダウン<br>オートマトン |
| 文脈依存文法<br>(1型文法) | 文脈依存言語<br>(1型言語) | 線形拘束<br>オートマトン |
| 0型文法 | 0型言語 | チューリング機械 |



# 自然言語処理に適した文法

- ×正規文法(=正規言語)
  - 自然言語を取り扱えない
- ×文脈依存文法, 0型文法
  - 文字列が文法によって生成できるかどうかを判定するのに膨大な計算時間を要する
- ○文脈自由文法
  - 自然言語を生成する十分な記述能力を持つ
  - 効率的なパージングアルゴリズム(parsing algorithm)が存在する



# まとめ

- 言語のモデル化
  - 言語を形式的に定義する
- 有限オートマトンと正規表現
  - 正規表現: 文字列の集合(言語)を定義
  - FA: 入力文字列が言語に属するかどうかを判定
- 形式文法
  - 文脈自由文法