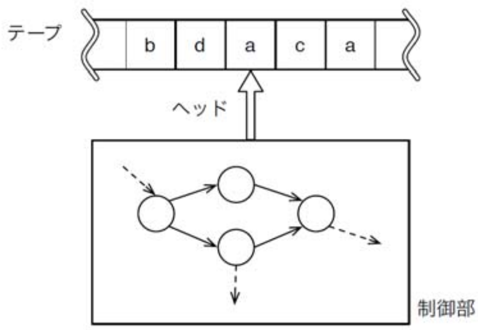
ソフトウェアモデル論（2011年度）
第3回・2011/10/14

桑原 寛明
情報理工学部 情報システム学科

オートマトンの概念図 （復習）



決定性有限オートマトン (DFA) （復習）

$M = (Q, \Sigma, \delta, q_0, F)$

Q : 状態の有限集合 $(\neq \emptyset)$

Σ : 入力記号の有限集合

δ : 状態遷移関数 $(Q \times \Sigma \to Q)$

q_0 : 初期状態 $(\in Q)$

F : 受理状態の集合 $(\subseteq Q)$

NFAでは2^Q

受理言語 （復習）

- $\delta(q_0,w) \in F$ならば語wは受理される
  - 開始状態から開始して受理状態で終了
- オートマトンMの受理言語 $L(M)=\{w \mid \delta(q_0,w) \in F\}$
  - 受理される語の集合
- $L(M_1)=L(M_2)$の時M_1とM_2は等しい
  - オートマトンが等しいとは受理言語が等しいこと

DFAとNFAの等価性 （復習）

- DFAはNFAの特殊な場合である
  - DFAは遷移先が一意に決められるNFA
- NFAをシミュレートするDFAを作ることができる
  - 受理言語が同じ
- 受理言語が同じであればNFAの方が作りやすいことが多い

NFAをシミュレートするDFAの例

NFA

$Q : \{S_0, S_1\}$

$\Sigma : \{a\}$

$\delta : (S_0, a) \to \{S_0, S_1\}$

$q_0 : S_0$

$F : \{S_1\}$

DFA

$Q : \{\emptyset, \{S_0\}, \{S_1\}, \{S_0, S_1\}\}$

$\Sigma : \{a\}$

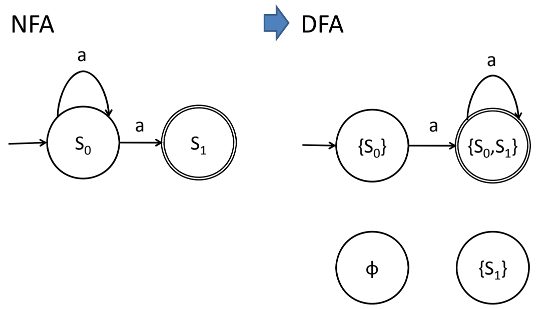
$\delta : (\{S_0\}, a) \to \{S_0, S_1\}$

$(\{S_0, S_1\}, a) \to \{S_0, S_1\}$

$q_0 : \{S_0\}$

$F : \{\{S_1\}, \{S_0, S_1\}\}$

NFAをシミュレートするDFAの例



NFAをシミュレートするDFAの作り方

- NFA : $M_N=(Q_N, \Sigma, \delta_N, q_0^N, F_N)$
- DFA : $M_D=(Q_D, \Sigma, \delta_D, q_0^D, F_D)$

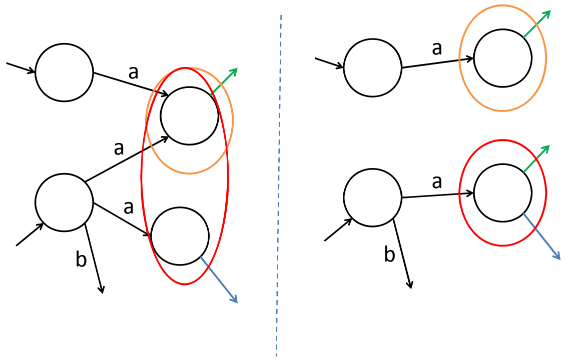
- M_Dの状態集合はM_Nの状態集合のべき集合
  - $Q_D = 2^{QN}$
- M_NとM_Dの記号の集合は同じ
- M_Dの初期状態はM_Nの初期状態のみからなる状態
  - $q_0^D = \{ q_0^N \}$

NFAをシミュレートするDFAの作り方

- M_Dの最終状態はM_Nの最終状態を1つでも含む状態すべて
  - $F_D = \{ q_D \mid q_D \in Q_D$ かつ $q_D \cap F_N \neq \varnothing \}$

- M_Dの各状態の遷移先は、その状態が含むM_Nの状態の遷移先すべてを含むM_Dの状態
  - $\delta_D(\{q_1^N,\ldots,q_i^N\}, a) = \delta_N(q_1^N,a) \cup \ldots \cup \delta_N(q_i^N,a)$

NFAをシミュレートするDFAの作り方



有限オートマトンで受理できない言語の例

- $\{ a^n b^n \mid n$は0以上の整数 $\}$
- $\{ a^j \mid j=i^2$ で i は1以上の整数 $\}$

- 有限オートマトンで受理できるのはどのような言語だろうか？

正規表現 (RE)

アルファベットΣ上の正規表現

1. 空列ε、空集合∅、記号 a∈Σ は正規表現
2. P, Qが正規表現ならば
   - (P + Q)　選択
   - (P • Q)　連接（通常は • を省略する）
   - (P*)　繰返し

   は正規表現
3. 1.と2. を有限回適用して生成されるものだけが正規表現

正規表現が表す語の集合

1. 正規表現 ε は集合 {ε} を表す
2. 正規表現 ∅ は空集合 ∅ を表す
3. 正規表現 a は集合 {a} を表す
4. 正規表現 P, Q が表す語の集合をそれぞれ ***P***, ***Q*** とすると
   - (P + Q) は $\{ w \mid w \in \boldsymbol{P} \lor w \in \boldsymbol{Q} \} = \boldsymbol{P} \cup \boldsymbol{Q}$
   - (P • Q) は $\{ v \bullet w \mid v \in \boldsymbol{P} \land w \in \boldsymbol{Q} \} = \boldsymbol{P} \bullet \boldsymbol{Q}$
   - (P*) は $\{\varepsilon\} \cup \boldsymbol{P} \cup \boldsymbol{P} \bullet \boldsymbol{P} \cup \ldots$

   を表す

正規表現と語の集合の例（Σ={a,b}）

正規表現	語の集合
a+b	{ a, b }
aabb	{ aabb }
a(a+b)b	{ aab, abb }
a*	{ ε, a, aa, aaa, aaaa, ... }
ab*a	{ aa, aba, abba, abbba, ... }
a(a+b)*b	{ ab, aab, abb, aaab, aabb, abab, abbb, aaaab, ... }

正規表現の変形規則

- R(S+T) = RS + RT
- (R+S)T = RT + ST
- R*(R+ε) = R* = (R+ε)R*
- $R^* + R^n = R^*$
- (R+ε)* = R*
- RS* + R = RS*
- ε* = ε
- Rε = R = εR
- R∅ = ∅ = ∅R
- R + ∅ = R = ∅ + R

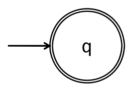
有限オートマトンと正規表現の等価性

- RE → NFA → DFA
  - REが表す言語を受理するNFAを作る
  - NFAをシミュレートするDFAを作る
- NFA → DFA → RE
  - NFAをシミュレートするDFAを作る
  - DFAをREに変換する

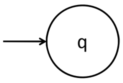
正規表現から有限オートマトンへの変換

1. 正規表現 ε、∅、a が表す言語を受理する有限オートマトンを作る
2. 正規表現 (P+Q)、(P•Q)、(P*) が表す言語を受理する有限オートマトンは、P や Q が表す言語を受理する有限オートマトンを組合せて作る

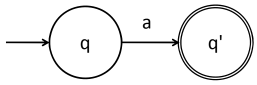
{ε} を受理する有限オートマトン



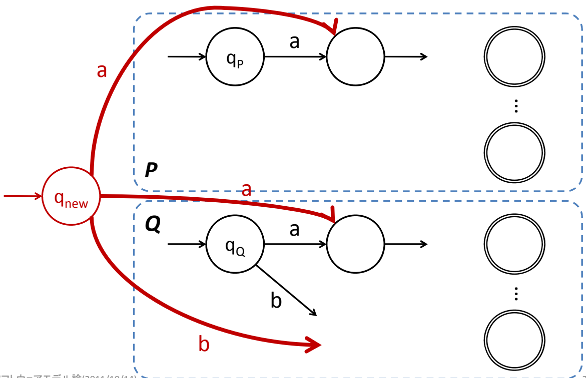
∅ を受理する有限オートマトン



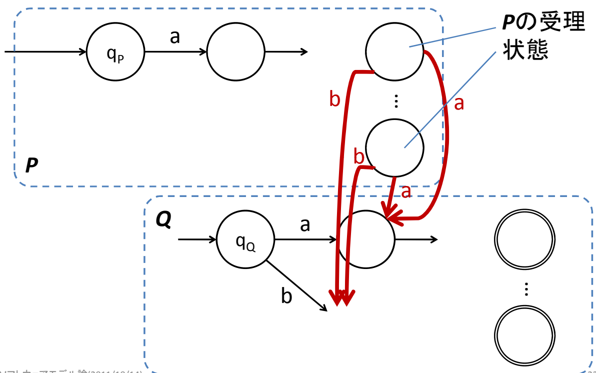
{a} を受理する有限オートマトン



P+*Q* を受理する有限オートマトン

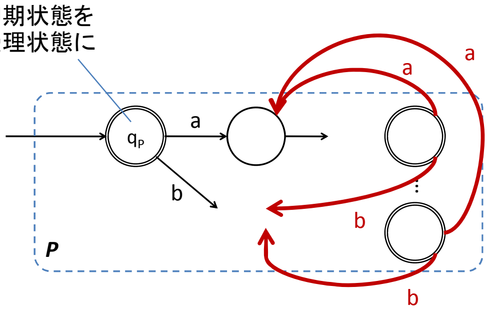


P•*Q* を受理する有限オートマトン

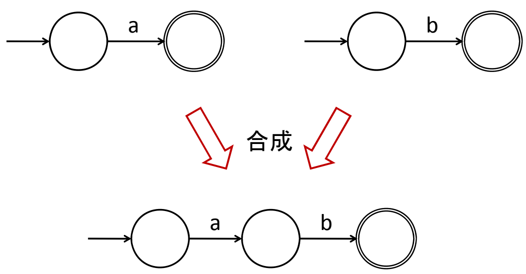


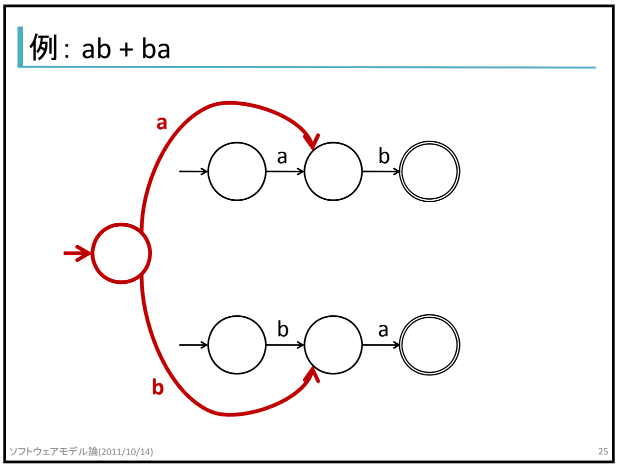
*P** を受理する有限オートマトン

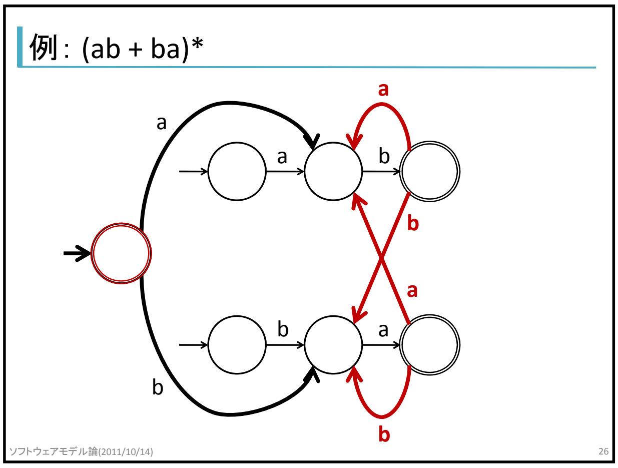
初期状態を
受理状態に



例：ab




例 : ab + ba
a
a
b
b
b
a
ソフトウェアモデル論(2011/10/14)
25


例 : (ab + ba)*
a
a
a
b
b
a
b
b
a
b
ソフトウェアモデル論(2011/10/14)
26