

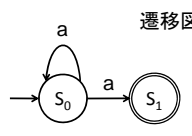
ソフトウェアモデル論(2012年度)
第4回・2012/10/19

桑原 寛明
情報理工学部 情報システム学科

NFAの例 (復習)

- a の1回以上の繰返しを受理するNFA

$Q : \{S_0, S_1\}$
 $\Sigma : \{a\}$
 $\delta : (S_0, a) \rightarrow \{S_0, S_1\}$
 $q_0 : S_0$
 $F : \{S_1\}$



遷移図

遷移表

| | a |
|----------------|---------------------------------|
| S ₀ | S ₀ , S ₁ |
| S ₁ | - |

ソフトウェアモデル論(2012/10/19) 2

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

ソフトウェアモデル論(2012/10/19) 3

NFAをシミュレートするDFAの作り方 (復習)

- 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
 - $Q_D = 2^{Q_N}$
- 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
 - $q_0^D = \{q_0^N\}$

ソフトウェアモデル論(2012/10/19) 4

NFAをシミュレートするDFAの作り方 (復習)

- 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
 - $F_D = \{q_D \mid q_D \in Q_D \text{ かつ } q_D \cap F_N \neq \emptyset\}$
- 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
 - $\delta_D(\{q_1^N, \dots, q_i^N\}, a) = \delta_N(q_1^N, a) \cup \dots \cup \delta_N(q_i^N, a)$

ソフトウェアモデル論(2012/10/19) 5

NFAをシミュレートするDFAの例 (復習)

| NFA | DFA |
|--|---|
| $Q : \{S_0, S_1\}$ | $Q : \{\emptyset, \{S_0\}, \{S_1\}, \{S_0, S_1\}\}$ |
| $\Sigma : \{a\}$ | $\Sigma : \{a\}$ |
| $\delta : (S_0, a) \rightarrow \{S_0, S_1\}$ | $\delta : (\emptyset, a) \rightarrow \emptyset$ |
| $q_0 : S_0$ | $(\{S_0\}, a) \rightarrow \{S_0, S_1\}$ |
| $F : \{S_1\}$ | $(\{S_1\}, a) \rightarrow \emptyset$ |
| | $(\{S_0, S_1\}, a) \rightarrow \{S_0, S_1\}$ |
| | $q_0 : \{S_0\}$ |
| | $F : \{\{S_1\}, \{S_0, S_1\}\}$ |

ソフトウェアモデル論(2012/10/19) 6

NFAをシミュレートするDFAの例 (復習)

ソフトウェアモデル論(2012/10/19) 7

正規表現 (RE) (復習)

アルファベット Σ 上の正規表現

- 空列 ϵ 、空集合 \emptyset 、記号 $a \in \Sigma$ は正規表現
- P, Q が正規表現ならば
 - $(P + Q)$ 選択
 - $(P \cdot Q)$ 接続(通常は \cdot を省略する)
 - (P^*) 繰返し
 は正規表現
- 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

ソフトウェアモデル論(2012/10/19) 8

正規表現が表す語の集合 (復習)

- 正規表現 ϵ は集合 $\{\epsilon\}$ を表す
- 正規表現 \emptyset は空集合 \emptyset を表す
- 正規表現 a は集合 $\{a\}$ を表す
- 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 $\{w \mid w \in P \vee w \in Q\} = P \cup Q$
 - $(P \cdot Q)$ は $\{v \cdot w \mid v \in P \wedge w \in Q\} = P \cdot Q$
 - (P^*) は $\{\epsilon\} \cup P \cup P \cdot P \cup \dots$ を表す

ソフトウェアモデル論(2012/10/19) 9

正規表現と語の集合の例 ($\Sigma=\{a,b\}$) (復習)

| | |
|---|--|
| <ul style="list-style-type: none"> 正規表現 <ul style="list-style-type: none"> $a+b$ $aabb$ $a(a+b)b$ a^* ab^*a $a(a+b)^*b$ | <ul style="list-style-type: none"> 語の集合 <ul style="list-style-type: none"> $\{a, b\}$ $\{aabb\}$ $\{aab, abb\}$ $\{\epsilon, a, aa, aaa, aaaa, \dots\}$ $\{aa, aba, abba, abbbb, \dots\}$ $\{ab, aab, abb, aaab, aabb, abab, abbb, aaaab, \dots\}$ |
|---|--|

ソフトウェアモデル論(2012/10/19) 10

有限オートマトンと正規表現の等価性 (復習)

- 正規表現で表現できる言語は有限オートマトンで受理できる
- 有限オートマトンが受理できる言語は正規表現で表現できる

ソフトウェアモデル論(2012/10/19) 11

有限オートマトンと正規表現の等価性 (復習)

- $RE \rightarrow NFA \rightarrow DFA$
 - 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
- $NFA \rightarrow DFA \rightarrow RE$
 - NFAをシミュレートするDFAを作る
 - DFAをREに変換する

ソフトウェアモデル論(2012/10/19) 12

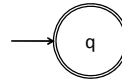
正規表現から有限オートマトンへの変換(復習)

1. 正規表現 ϵ , \emptyset , 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 $(P+Q)$, $(P \cdot Q)$, (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

ソフトウェアモデル論(2012/10/19)

13

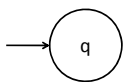
$\{\epsilon\}$ を受理する有限オートマトン (復習)



ソフトウェアモデル論(2012/10/19)

14

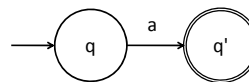
\emptyset を受理する有限オートマトン (復習)



ソフトウェアモデル論(2012/10/19)

15

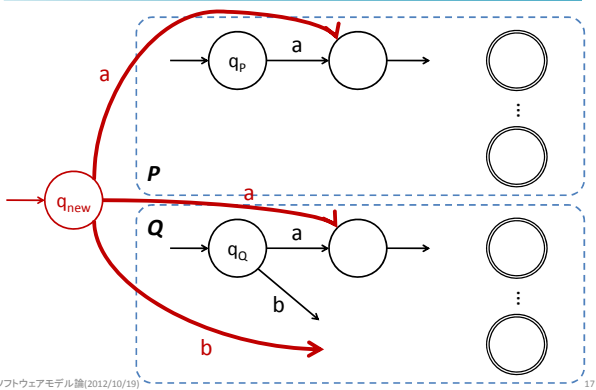
$\{a\}$ を受理する有限オートマトン (復習)



ソフトウェアモデル論(2012/10/19)

16

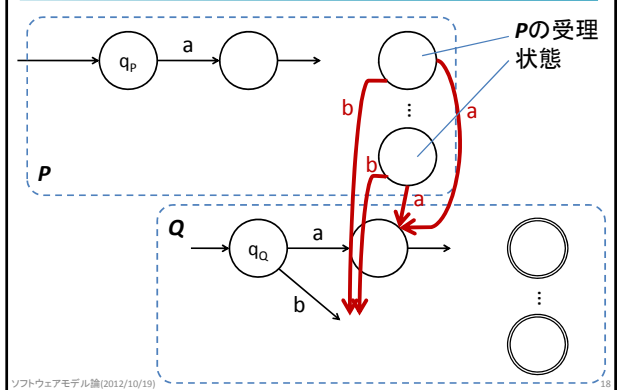
$P+Q$ を受理する有限オートマトン



ソフトウェアモデル論(2012/10/19)

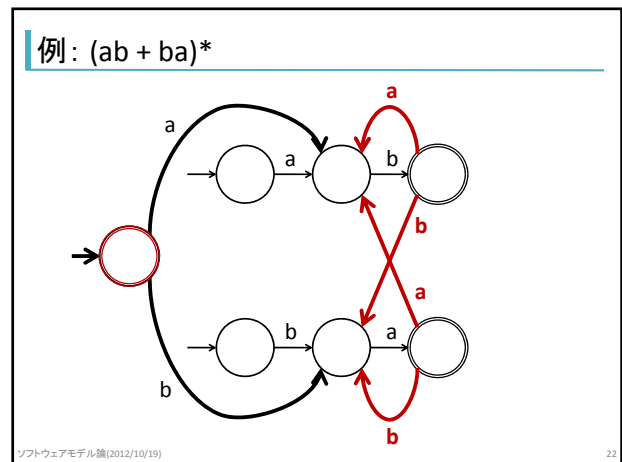
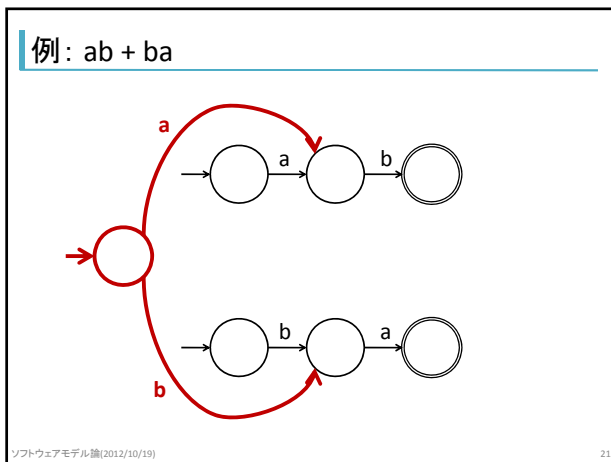
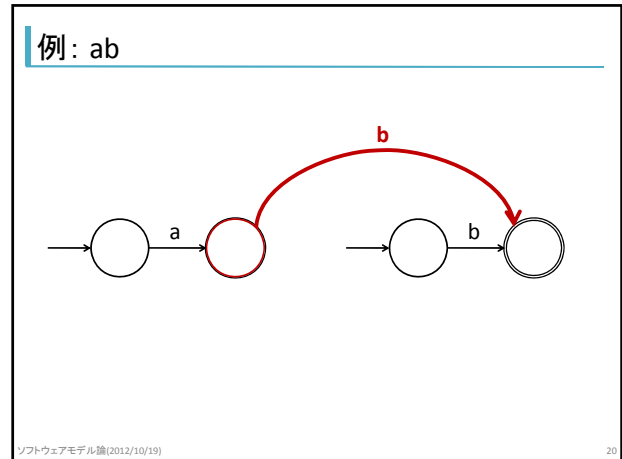
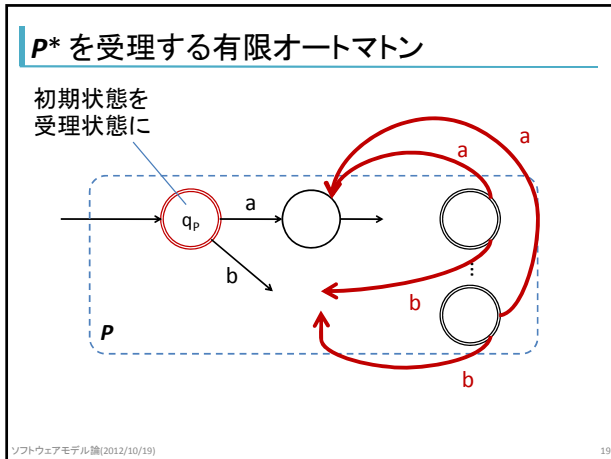
17

$P \cdot Q$ を受理する有限オートマトン



ソフトウェアモデル論(2012/10/19)

18



有限オートマトンから正規表現への変換

- 有限オートマトン $M = (\{1, \dots, n\}, \Sigma, \delta, 1, F)$
- r_{ij}^k を求める
 - 状態 i から状態 j へ k 以下の状態のみを通過して到達する記号列の正規表現
 - r_{ij}^0 から順に帰納的に
- $r_{1f1}^n + \dots + r_{1fl}^n$ が初期状態から受理状態へ到達する記号列の正規表現
 - $\{f1, \dots, fl\}$ は受理状態の集合

ソフトウェアモデル論(2012/10/19) 23

r_{ij}^0 を求める

- 状態 i から状態 j へ直接到達
- $\delta(i, a) = j$ を満たす a の集合を $\{a_1, \dots, a_l\}$ とする
 - そのような a がなければ \emptyset
- $i \neq j$ ならば $r_{ij}^0 = a_1 + \dots + a_l$
- $i = j$ ならば $r_{ij}^0 = a_1 + \dots + a_l + \epsilon$

ソフトウェアモデル論(2012/10/19) 24

r_{ij}^k を求める

- $r_{ij}^k = r_{ik}^{k-1} (r_{kk}^{k-1}) * r_{kj}^{k-1} + r_{ij}^{k-1}$
- 状態 k を通過するかないかの2択
 - 通過する
 1. i から $k-1$ 以下を通過して初めて k に到達
 2. $k-1$ 以下のみ通過してまた k に到達(を繰り返す)
 3. k から $k-1$ 以下を通過して j に到達
 - 通過しない
 - $k-1$ 以下の状態のみ通過