コンパイラ

第3回 字句解析
― 決定性有限オートマトンの導出 ―

http://www.info.kindai.ac.jp/compiler

E館3階E-331 内線5459

takasi-i@info.kindai.ac.jp

コンパイラの構造

- 字句解析系
- 構文解析系
- 制約検査系
- 中間コード生成系
- 最適化系
- 目的コード生成系

処理の流れ
情報システムプロジェクトIの場合

output (ab);

↓

字句解析系　　マイクロ構文の文法に従い解析

↓

“output” “(” 変数名 “)” “;”

↓

構文解析系　　マクロ構文の文法に従い解析

↓

<output_statement> ::= “output” “(” <exp> “)” “;”

↓

コード生成系　　VSMアセンブラの文法に従い生成

↓

1. PUSH &ab　　2. OUTPUT

字句解析系 (lexical analyzer, scanner)

- 字句解析系
  - 空白、コメントを読み飛ばす
  - 単語(token)に区切る
  - マイクロ構文エラーを検出

```
if (ans >= 123 ) /* ansの値で分岐 */ (改行)
    (空白)output (‘1’) ;
```

予約語 “if”
左括弧 “(”
変数　“ans”
不等号 “>=”
整数　“123”
右括弧 “)”
予約語 “output”
　：

単語への分割

英語の場合

School of Science and Engineering Kindai University

⇒単語間に空白があるので区切るのは簡単

日本語の場合

きんきだいがくりこうがくぶ

きんき : だいがく : りこうがくぶ　近畿 大学 理工学部

きんきだ : いがくり : こうが : くぶ　近畿だ イガ栗 黄河 九分

区切り方を正しく決定するのは困難

計算機言語の場合は？

単語への分割

計算機言語の場合

区切り記号で単語を判別できる

```
main () {
    int i, j, k;
        :
```

m a i n (

区切り記号 (が来たので
“main” で区切ると判別

マイクロ構文
(情報システムプロジェクトIの場合)

- マイクロ構文 (EBNF記法で定義)

または　0回以上の繰り返し

K-Program ::= { Token | W-Space } ‘¥0’(ファイル末)

変数名　整数

Token(単語) ::= NAME | INTEGER

文字　文字列

| CHARACTER | STRING

| KEYWORD | OPERATOR | DELIMITER

予約語　演算子　区切り記号

W-Space(空白) ::= ‘ ’(スペース) | ‘¥t’ (タブ) | ‘¥n’ (改行)

| Comment

コメント

(文字列，コメントは拡張課題)

マイクロ構文(変数名, 整数, 文字)

- マイクロ構文

NAME(変数名) ::= Alpha { Alpha | Dec }
INTEGER(整数) ::= ‘0’ | Pdec { Dec }
| ‘0’ ‘x’ Xdec { Xdec }
CHARACER(文字) ::= ‘‘(シングルクォート)’ Character ‘’’
STRING(文字列) ::= ‘“(ダブルクォート)’ { Character } ‘”’

Alpha ∈ {a, b, …, z, A, B, …, Z, _(アンダーバー)}
Pdec ∈ {1, 2, …, 9}
Dec ∈ {0, 1, 2, …, 9}
Xdec ∈ {0, 1, 2, …, 9, A, B, …, F}
Character ::= (任意の文字)

マイクロ構文(予約語)

- マイクロ構文

KEYWORD(予約語) ::= ‘f’ ‘o’ ‘r’
| ‘i’ ‘f’
| ‘i’ ‘n’ ‘t’
| ‘i’ ‘n’ ‘p’ ‘u’ ‘t’ ‘c’ ‘h’ ‘a’ ‘r’
| ‘i’ ‘n’ ‘p’ ‘u’ ‘t’ ‘i’ ‘n’ ‘t’
| ‘m’ ‘a’ ‘i’ ‘n’
| ‘o’ ‘u’ ‘t’ ‘p’ ‘u’ ‘t’ ‘c’ ‘h’ ‘a’ ‘r’
| ‘o’ ‘u’ ‘t’ ‘p’ ‘u’ ‘t’ ‘i’ ‘n’ ‘t’
| ‘o’ ‘u’ ‘t’ ‘p’ ‘u’ ‘t’ ‘s’ ‘t’ ‘r’
| ‘s’ ‘e’ ‘t’ ‘s’ ‘t’ ‘r’
| ‘w’ ‘h’ ‘i’ ‘l’ ‘e’

予約語は変数名では使用不可

拡張課題では ‘e’ ‘l’ ‘s’ ‘e’ 等も予約語

マイクロ構文(演算子, 区切り記号)

- マイクロ構文

OPERATOR(演算子) ::=

‘=’ ‘=’ | ‘!’ ‘=’ | ‘<’ | ‘>’

| ‘&’ ‘&’ | ‘|’ ‘|’ | ‘!’

| ‘+’ | ‘-’ | ‘*’ | ‘/’ | ‘%’

| ‘=’ | ‘+’ ‘=’ | ‘-’ ‘=’ | ‘*’ ‘=’ | ‘/’ ‘=’

| ‘+’ ‘+’ | ‘-’ ‘-’

DELIMITER(区切り記号) ::=

‘;’ | ‘,’ | ‘(’ | ‘)’ | ‘{’ | ‘}’ | ‘[’ | ‘]’

トークンの種類
(情報システムプロジェクトIの場合)

トークン		記号
区切り記号		; , ( ) { } [ ]
演算子	比較演算子	== != < > (<=) (>=)
	論理演算子	! && ||
	算術演算子	+ - * / %
	代入演算子	= += -= *= /= ++ --
名前		変数名
定数		整数 文字 文字列
予約語		main int if while for inputint inputchar outputint outputchar outputstr setstr (else) (do) (break) …

トークン名

区切り記号

記号	トークン名
;	SEMICOLON
,	COMMA
(	LPAREN
)	RPAREN
{	LBRACE
}	RBRACE
[	LBRACKET
]	RBRACKET

比較演算子

記号	トークン名
==	EQUAL
!=	NOTEQ
<	LESS
>	GREAT
(<=)	LESSEQ
(>=)	GREATEQ

論理演算子

記号	トークン名
&&	AND
||	OR
!	NOT

トークン名

算術演算子

記号	トークン名
+	ADD
-	SUB
*	MUL
/	DIV
%	MOD

(※)

代入演算子

記号	トークン名
=	ASSIGN
+=	ASSIGNADD
-=	ASSIGNSUB
*=	ASSIGNMUL
/=	ASSIGNDIV
(%=)	ASSIGNMOD
++	INC
--	DEC

(※) 単項演算子の - と
二項演算子の - で
共通して使用

トークン名

定数, 変数名, 予約語

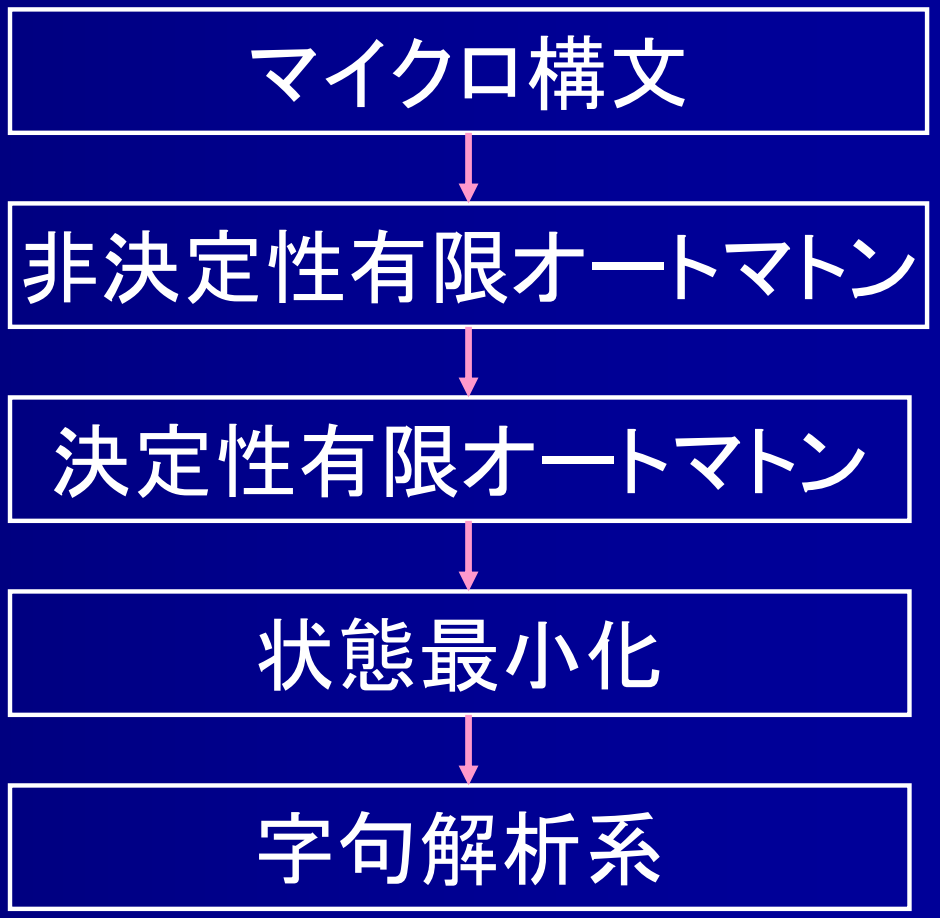
文字列	種別	トークン名
数字の並び	整数	INTEGER
‘(任意の1文字)’	文字	CHARACTER
“(任意の文字列)”	文字列	STRING
英字の並び	変数名	NAME
main	予約語	MAIN
int	予約語	INT
if	予約語	IF
while	予約語	WHILE
inputint	予約語	INPUTINT
:	:	:

INTEGER と INT の違いに注意

字句解析の手順

- 字句解析
  - 決定性有限オートマトンで解析

⇒(最小の)決定性有限オートマトンを
導出する必要がある



非決定性有限オートマトンへ

- マイクロ構文

→非決定性有限オートマトン(NFA)

以下の手法で帰納的に

1. ε(空記号列)に対するNFA
2. φ(空遷移関数集合)に対するNFA
3. a ∈Σ に対するNFA
4. $R_1 | R_2$ に対するNFA
5. R_1R_2 に対するNFA
6. R? に対するNFA
7. R* に対するNFA
8. R^+ に対するNFA　　(R_1, R_2, R：正規表現)

非決定性有限オートマトンへ

1. ε(空記号列)に対するNFA

入力が無くても
状態遷移

2. φ(空遷移関数集合)に対するNFA

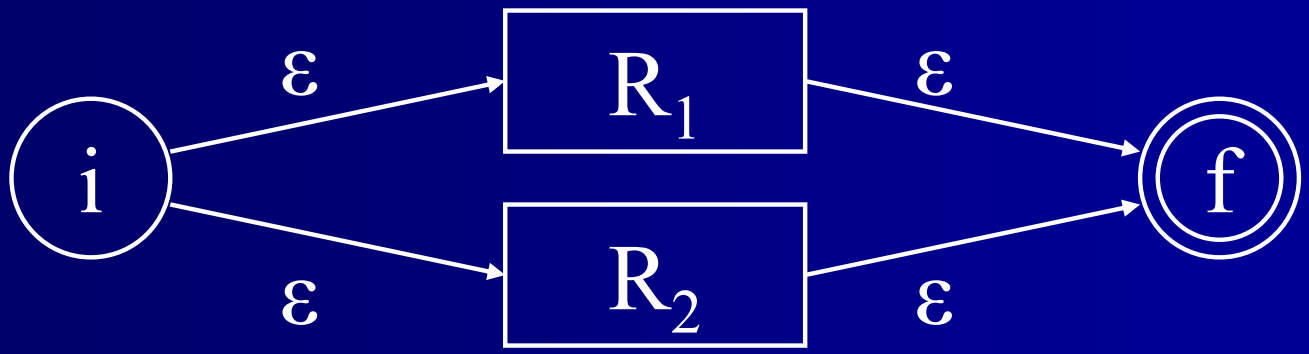
状態遷移無し

3. a ∈Σ に対するNFA

入力 a で
状態遷移

非決定性有限オートマトンへ

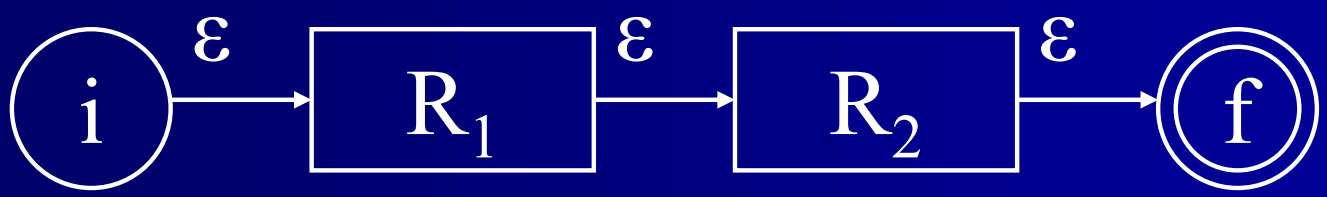
4. $R_1 \mid R_2$ に対するNFA



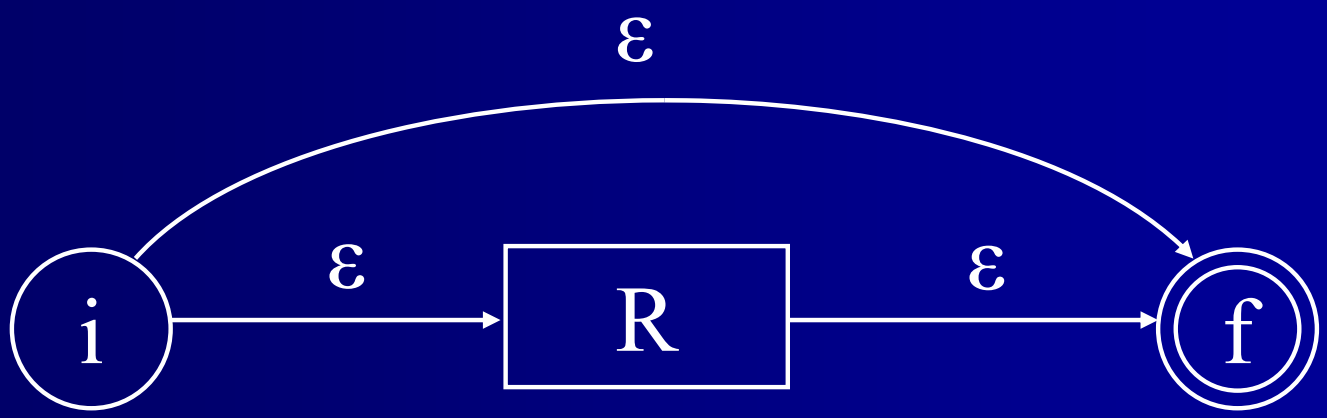
(R_1, R_2：正規表現)

非決定性有限オートマトンへ

5. R_1R_2 に対するNFA



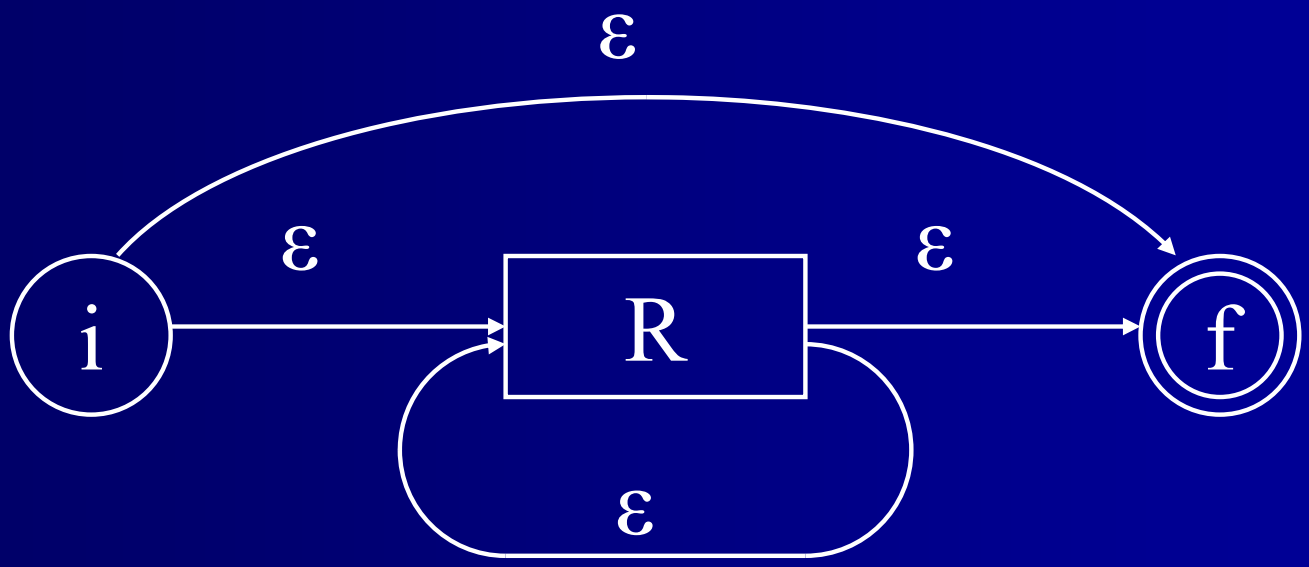
6. R? (省略, 0回または1回) に対するNFA



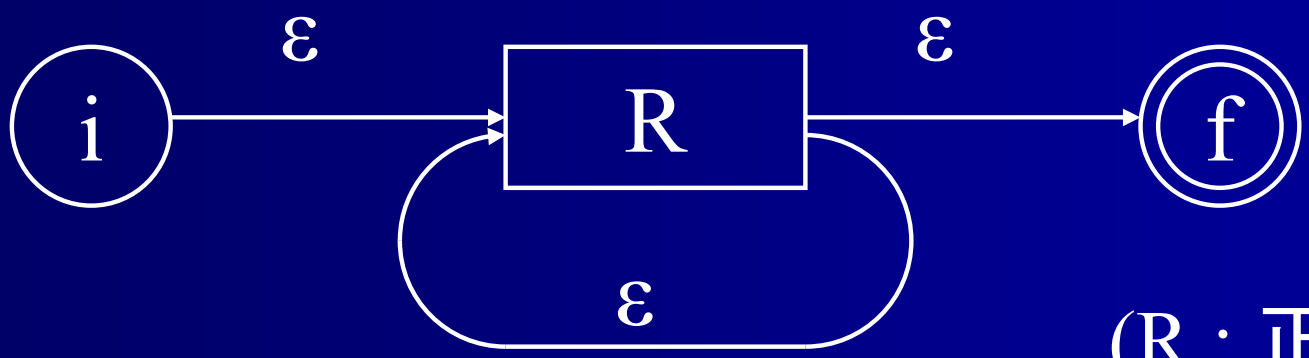
(R, R_1, R_2 : 正規表現)

非決定性有限オートマトンへ

7. R^* (0回以上の繰り返し) に対するNFA



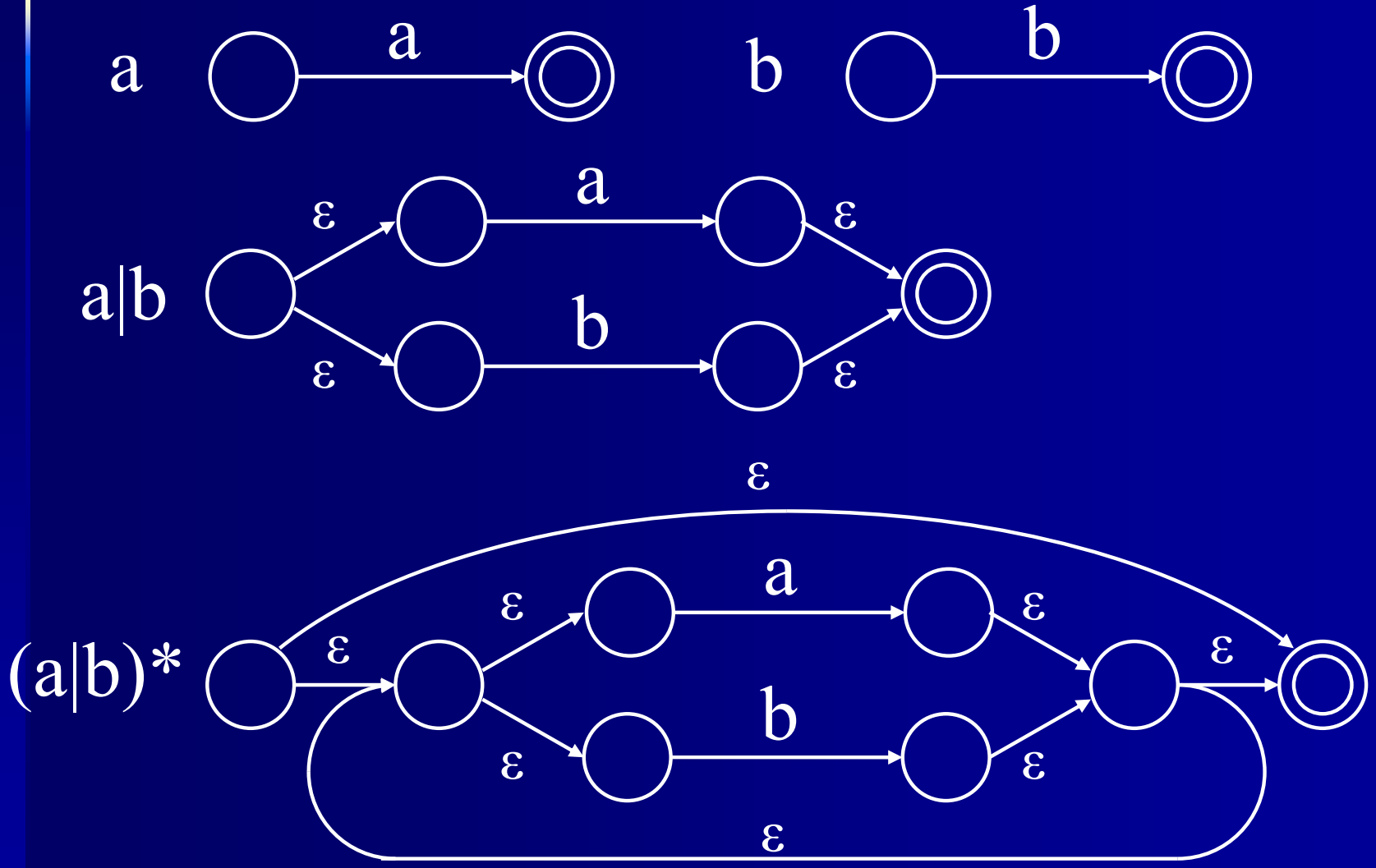
8. R^+ (1回以上の繰り返し) に対するNFA



(R：正規表現)

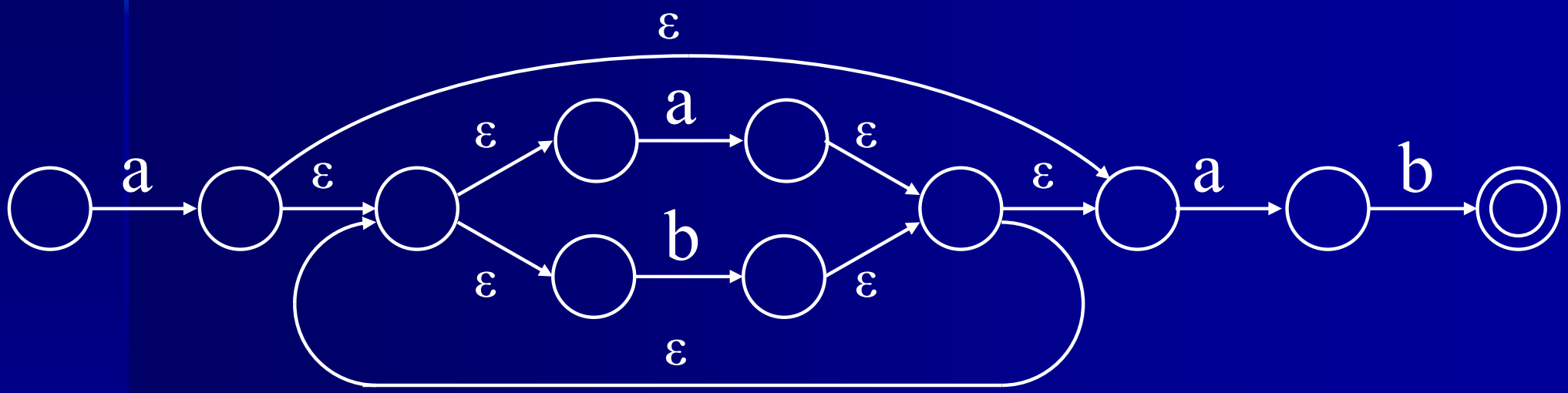
非決定性オートマトンへ

r ::= a (a|b)* a b



非決定性オートマトンへ

r ::= a (a|b)* a b

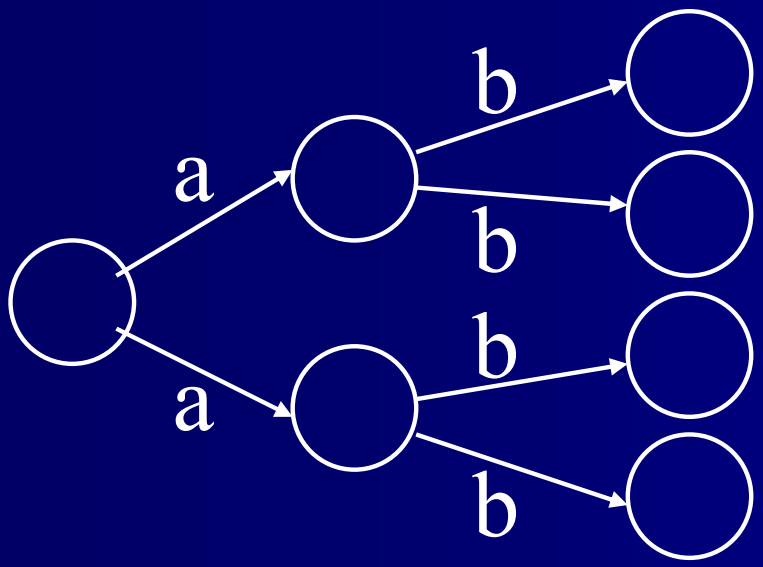


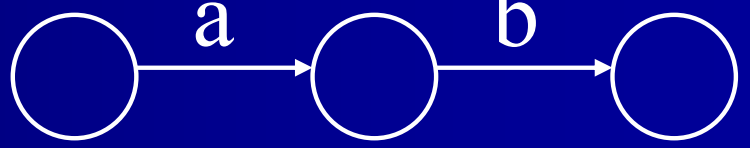
非決定性有限オートマトンの問題点

- 非決定性有限オートマトン
  - 同一入力に対する状態遷移が複数存在

  ⇒複数の遷移を全て解析する必要がある

  ⇒決定性有限オートマトンに変形する



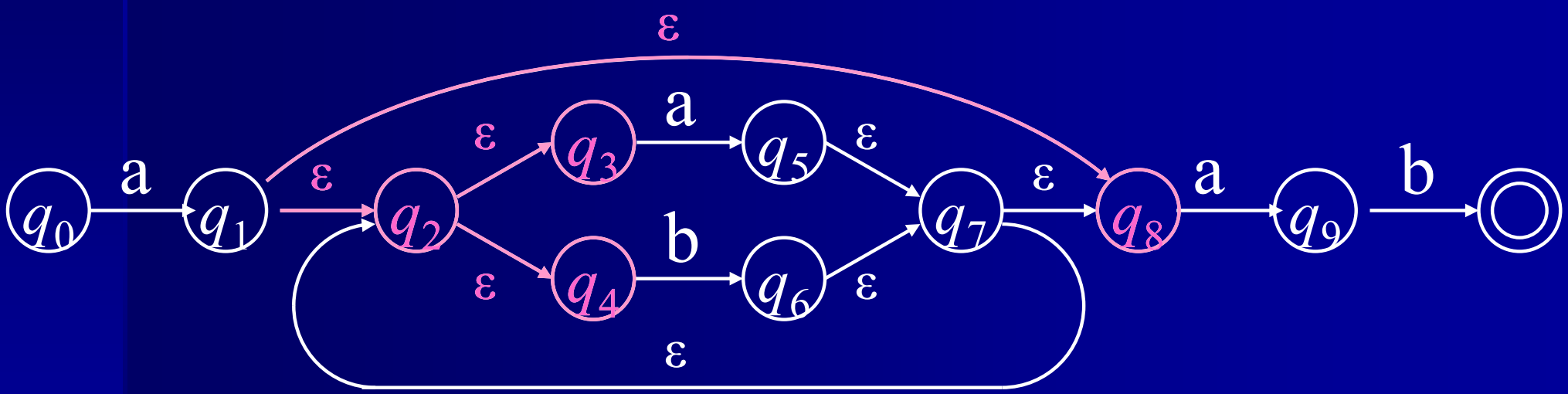


決定性有限オートマトンへ

- 非決定性有限オートマトン(NFA)

→決定性有限オートマトン(DFA)

同一の入力で遷移できる状態を1つの状態にまとめる



q_1 からε入力で $q_2\ q_3\ q_4\ q_8$ へ遷移可能

$\Rightarrow q_1\ q_2\ q_3\ q_4\ q_8$ をまとめる

決定性有限オートマトンへ

- NFA $N = (\mathbf{Q}_N, \mathbf{\Sigma}, \delta_N, q_{N0}, \mathbf{F}_N)$

$\Rightarrow$ DFA $D = (\mathbf{Q}_D, \mathbf{\Sigma}\ \delta_D, q_{D0}, \mathbf{F}_D)$

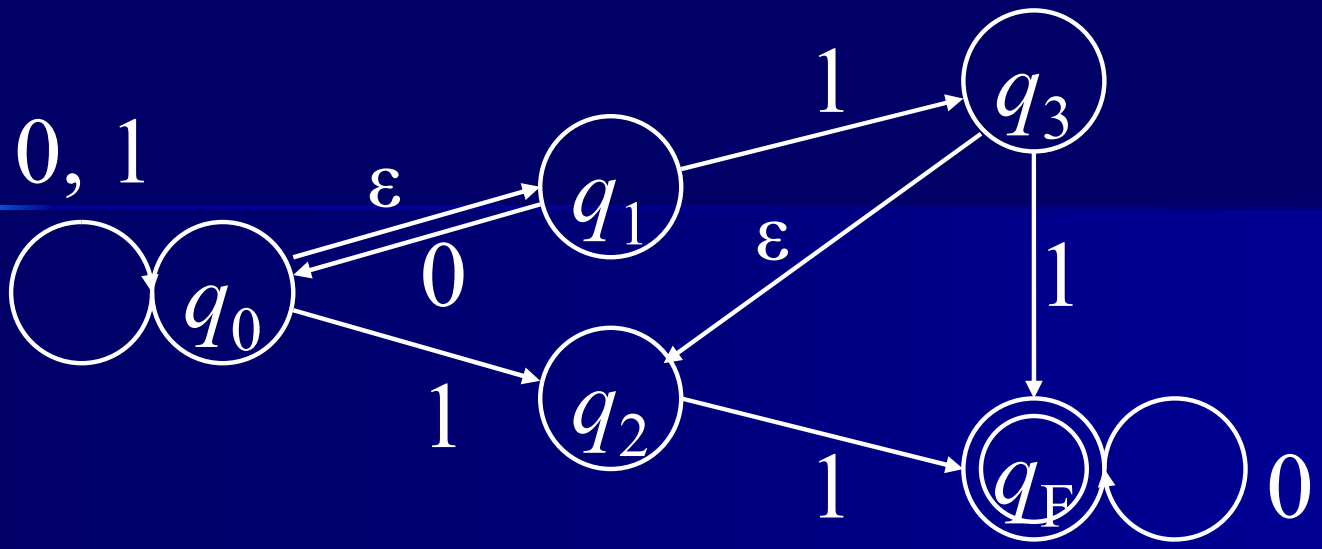
ε-closure (q)

::= $q \in \{\mathbf{Q}_N \cup \mathbf{Q}_D\}$ からε遷移できる状態集合

goto (q, a)

::= $q \in \{\mathbf{Q}_N \cup \mathbf{Q}_D\}$ から a $\in \mathbf{\Sigma}$で遷移できる状態集合

決定性有限オートマトンへ

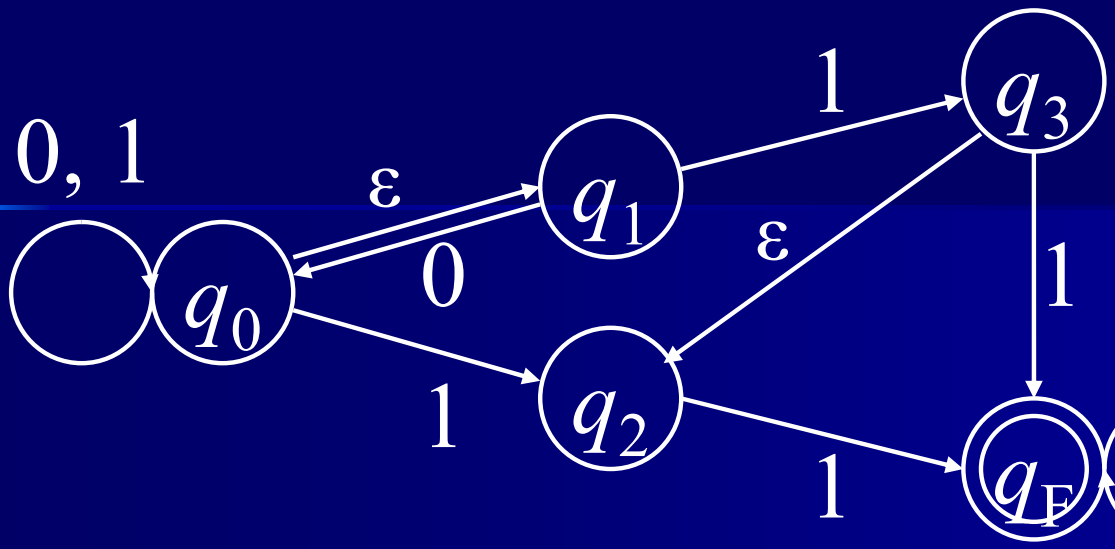


NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$		
q_1	$\{q_1\}$		
q_2	$\{q_2\}$		
q_3	$\{q_2, q_3\}$		
q_F	$\{q_F\}$		

q_0 のε-遷移先

決定性有限オートマトンへ



1　- 遷移は
ε1　- 遷移
1ε　- 遷移
ε1ε - 遷移を含む

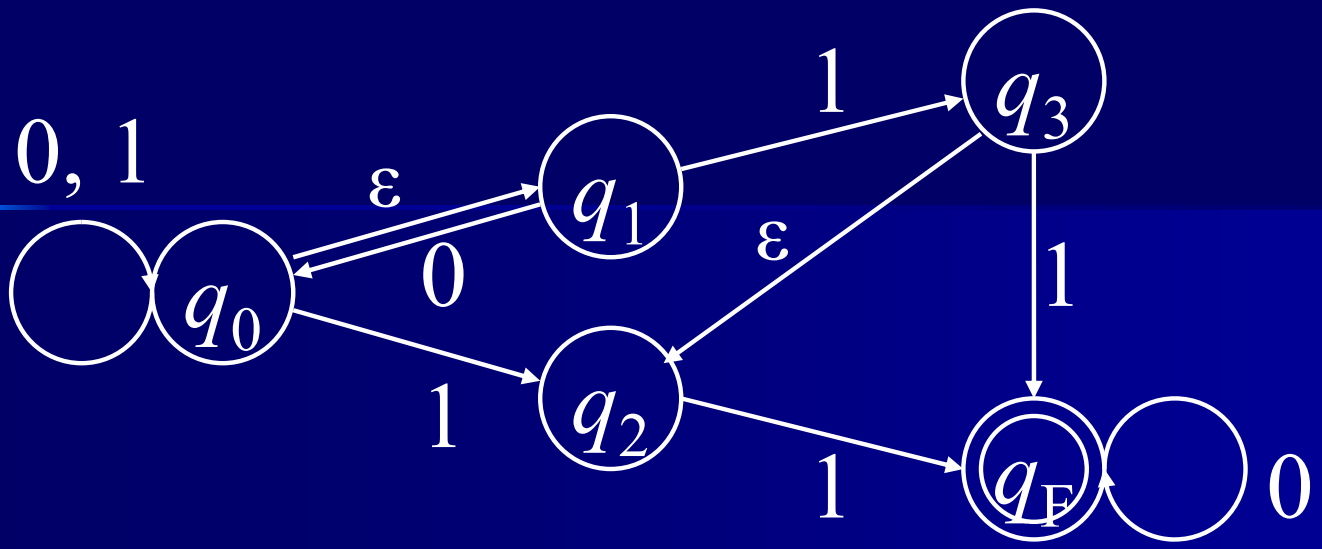
NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$		
q_2	$\{q_2\}$		
q_3	$\{q_2, q_3\}$		
q_F	$\{q_F\}$		

$\{q_0, q_1\}$の
0入力遷移先

$\{q_0, q_1\}$の
1入力遷移先

決定性有限オートマトンへ



NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_2	$\{q_2\}$	φ	$\{q_F\}$
q_3	$\{q_2, q_3\}$	φ	$\{q_F\}$
q_F	$\{q_F\}$	$\{q_F\}$	φ

決定性有限オートマトンへ

- **NFA⇒DFAアルゴリズム**

```
Q_D := {ε-closure (q_N0) };  /* Q_D の初期入力 */
for each q∈Q_D {             /* Q_D内の未実行の要素に対して実行*/
    for each a∈Σ {
        r := ε-closure (goto (q, a));
        if (r ∉ Q_D)         /* r はQ_Dの中に無いか？ */
            Q_D := Q_D ∪ r;  /* r をQ_Dに加える */
        δ_D(q, a) := r;
    }
}
```

決定性有限オートマトンへ

NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_2	$\{q_2\}$	φ	$\{q_F\}$
q_3	$\{q_2, q_3\}$	φ	$\{q_F\}$
q_F	$\{q_F\}$	$\{q_F\}$	φ

DFA

Q_D	ε-closure (q)	goto (q, 0)	goto (q, 1)
$q_{0,1}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$

ε-closure ($\{q_0 \cup q_1\}$)

goto ($\{q_0 \cup q_1\}$, 0)

goto ($\{q_0 \cup q_1\}$, 1)

決定性有限オートマトンへ

NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_2	$\{q_2\}$	φ	$\{q_F\}$
q_3	$\{q_2, q_3\}$	φ	$\{q_F\}$
q_F	$\{q_F\}$	$\{q_F\}$	φ

DFA

$q_{0,1} \in Q_D$

$q_{0,1} \in Q_D$

$q_{0,1,2,3} \notin Q_D$

Q_D	ε-closure (q)	goto (q, 0)	goto (q, 1)
$q_{0,1}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
$q_{0,1,2,3}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3, q_F\}$

決定性有限オートマトンへ

NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_2	$\{q_2\}$	φ	$\{q_F\}$
q_3	$\{q_2, q_3\}$	φ	$\{q_F\}$
q_F	$\{q_F\}$	$\{q_F\}$	φ

DFA

$q_{0,1} \in Q_D$

$q_{0,1,2,3,F} \notin Q_D$

Q_D	ε-closure (q)	goto (q, 0)	goto (q, 1)
$q_{0,1}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
$q_{0,1,2,3}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3, q_F\}$
$q_{0,1,2,3,F}$	$\{q_0, q_1, q_2, q_3, q_F\}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_2, q_3, q_F\}$

決定性有限オートマトンへ

NFA

Q_N	ε-closure (q)	goto (q, 0)	goto (q, 1)
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_2, q_3\}$
q_2	$\{q_2\}$	φ	$\{q_F\}$
q_3	$\{q_2, q_3\}$	φ	$\{q_F\}$
q_F	$\{q_F\}$	$\{q_F\}$	φ

DFA

Q_D	ε-closure (q)	goto (q, 0)	goto (q, 1)
$q_{0,1}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
$q_{0,1,2,3}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3, q_F\}$
$q_{0,1,2,3,F}$	$\{q_0, q_1, q_2, q_3, q_F\}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_2, q_3, q_F\}$
$q_{0,1,F}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_2, q_3\}$

$q_{0,1,F} \notin Q_D$

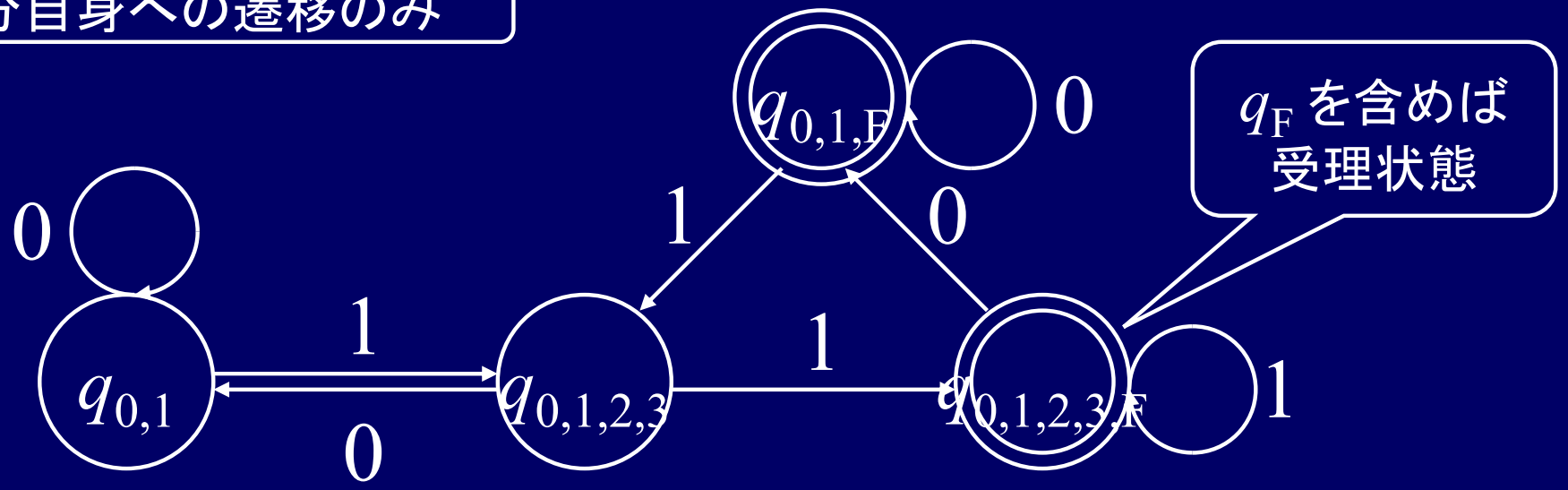
$q_{0,1,2,3,F} \in Q_D$

決定性有限オートマトンへ

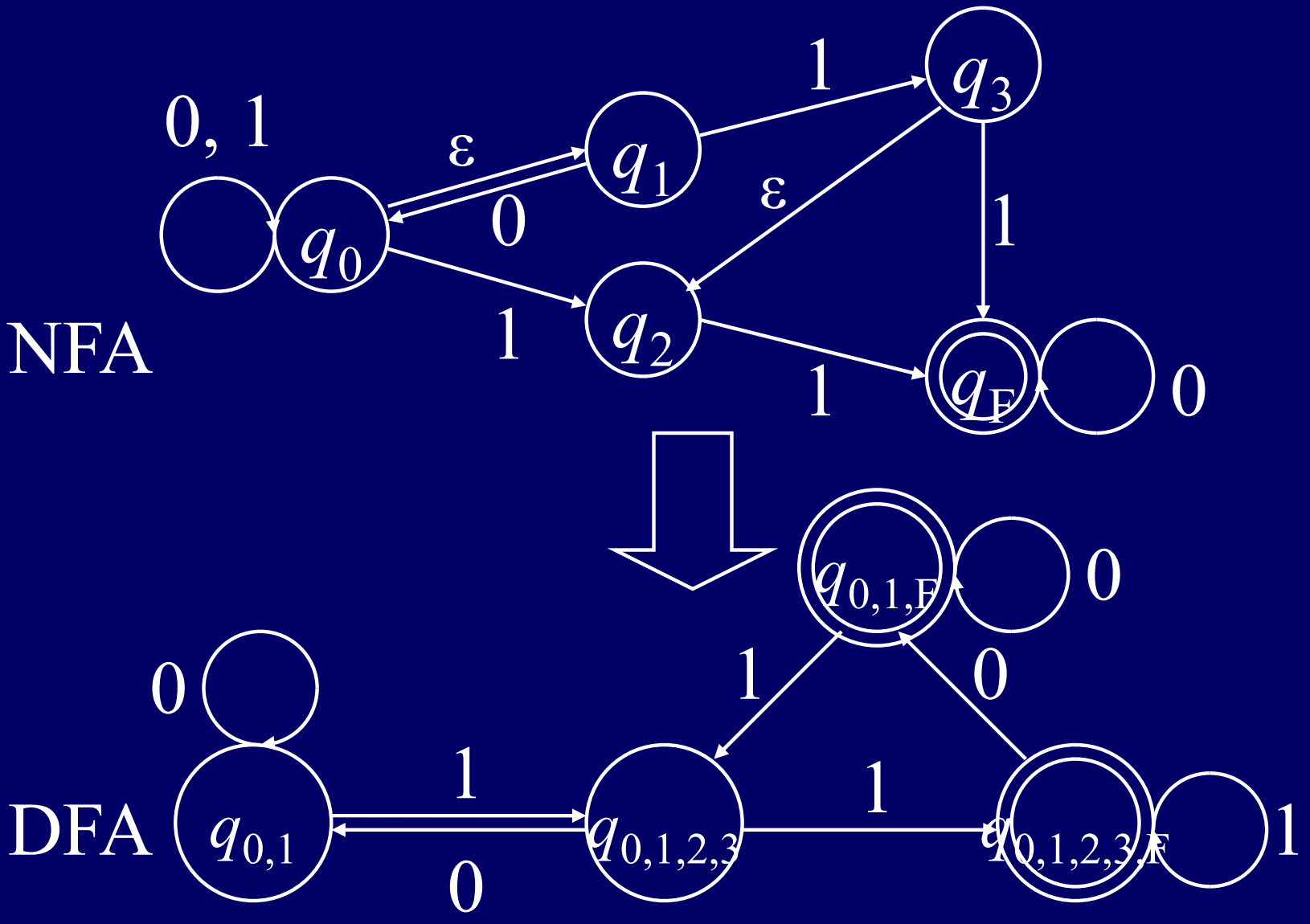
DFA

Q_D	ε-closure (q)	goto (q, 0)	goto (q, 1)
$q_{0,1}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3\}$
$q_{0,1,2,3}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2, q_3, q_F\}$
$q_{0,1,2,3,F}$	$\{q_0, q_1, q_2, q_3, q_F\}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_2, q_3, q_F\}$
$q_{0,1,F}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_F\}$	$\{q_0, q_1, q_2, q_3\}$

εは自分自身への遷移のみ



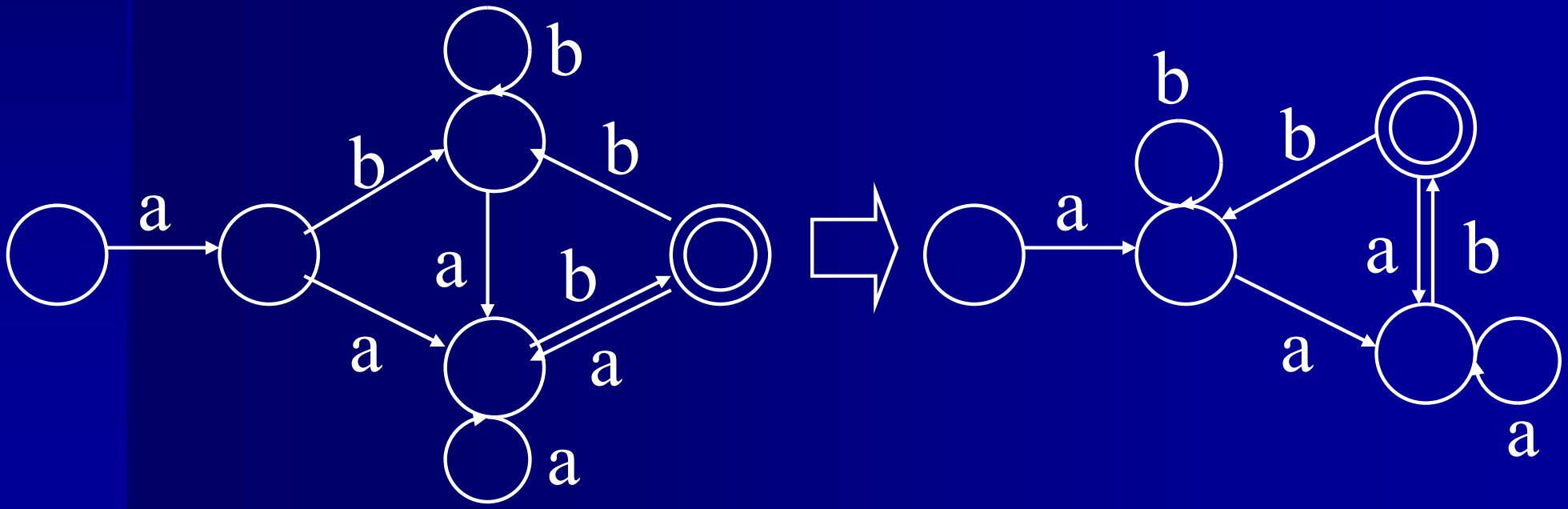
決定性有限オートマトンへ



決定性有限オートマトンの問題点

- 導出で得られた有限オートマトン
  - 状態数が最小とは限らない

  ⇒状態数の最小化を行う



状態最小化

- 状態最小化
  - 状態の等価性を用いてDFAを最適化

- 状態の等価性
  - 状態 p と状態 q に対して同一の入力列を与えたとき、その出力(受理, 不受理)が全て同じ

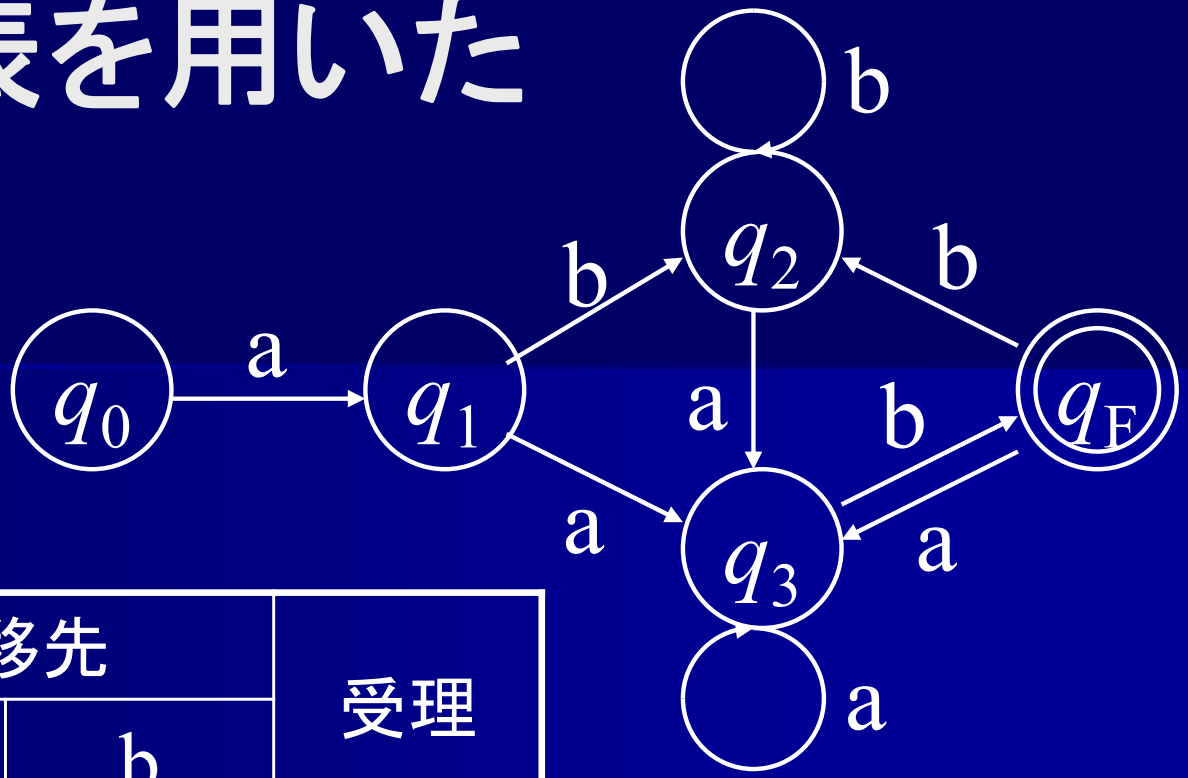
  ⇒状態 p と状態 q が等価である ($p \equiv q$)

(※) 等価性についての詳細は「論理回路」第13回講義を参照

状態数最小化の手順

- 手法1 状態遷移表の分割
  1. 異なる出力を生成する状態対をグループに分割
  2. 以下を分割できなくなるまで繰り返す
     - 同一の入力に対し、遷移先の状態が異なるグループに属すればその状態対をグループに分割
  3. グループごとに1つの状態に併合
- 手法2 状態併合表
  1. 異なる出力を持つ状態対に×を付ける
  2. 遷移先の状態対を記入
  3. 以下を×が付かなくなくなるまで繰り返す
     - 遷移先に×が付いていればその状態対に×を付ける
  4. 等価な状態対を決定

状態遷移表を用いた最小化



グループ $R^{(1)}$	状態 Q	遷移先		受理
		a	b	
0	0	1		
1	1	3	2	
1	2	3	2	
1	3	3	F	
2	F	3	2	○

遷移先, 受理でグループ分け

$\{q_0\}$
$\{q_1, q_2, q_3\}$
$\{q_F\}$

状態遷移表を用いた最小化

グループ $R^{(1)}$	状態 Q	遷移先		遷移先グループ		受理
		a	b	a	b	
0	0	1		1		
1	1	3	2	1	1	
	2	3	2	1	1	
	3	3	F	1	2	
2	F	3	2	1	1	○

q_1, q_2 と q_3 の
b入力遷移先グループが異なる

$\Rightarrow r_1$ を $\{q_1, q_2\}\{q_3\}$ に分割

$\{q_0\}$
$\{q_1, q_2\}$
$\{q_3\}$
$\{q_F\}$

状態遷移表を用いた最小化

グループ $R^{(2)}$	状態 Q	遷移先		遷移先グループ		受理
		a	b	a	b	
0	0	1		1		
1	1	3	2	2	1	
	2	3	2	2	1	
2	3	3	F	2	3	
3	F	3	2	2	1	○

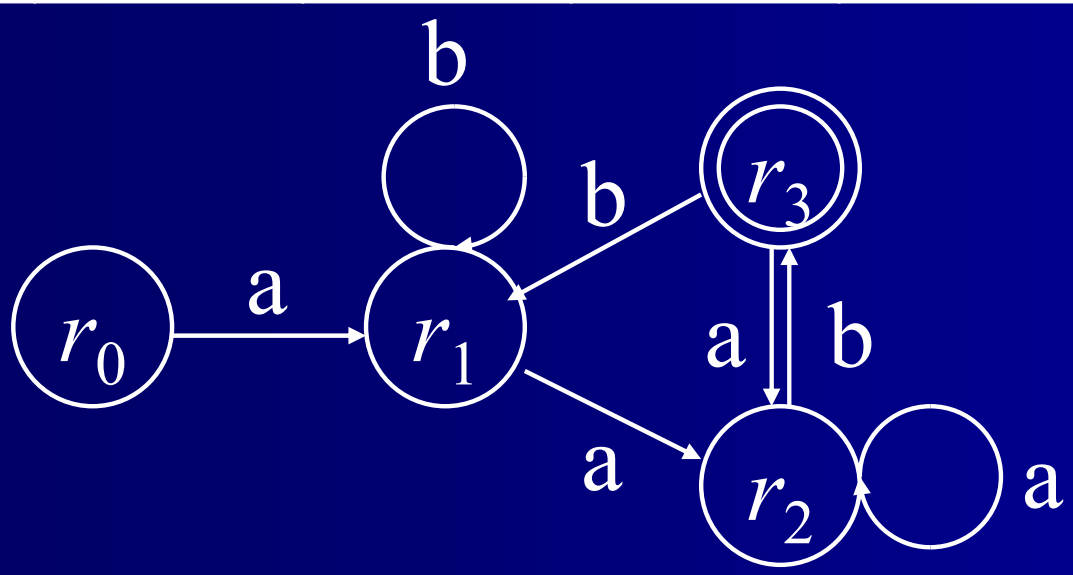
q_1, q_2 の遷移先グループが全て同じ

⇒ 分割終了

$\{q_0\}$
$\{q_1, q_2\}$
$\{q_3\}$
$\{q_F\}$

状態遷移表を用いた最小化

グループ $R^{(2)}$	状態 Q	遷移先グループ		受理
		a	b	
0	0	1		
1	1,2	2	1	
2	3	2	3	
3	F	2	1	○



状態併合表を用いた最小化

Q	遷移先		受理
	a	b	
0	1		
1	3	2	
2	3	2	
3	3	F	
F	3	2	○

0	
1	
2	
3	

q_0とq_1が異なるグループなら×を付ける

最後まで×が付かなければ等価

状態併合表を用いた最小化

Q	遷移先		受理				
	a	b					
0	1			0			
1	3	2		×	1		
2	3	2		×		2	
3	3	F		×			3
F	3	2	○	×	×	×	×

遷移先の有無、
受理不受理が
異なる状態対に
×を付ける

状態併合表を用いた最小化

Q	遷移先		受理				
	a	b					
0	1			0			
1	3	2		×	1		
2	3	2		×	3 3 2 2	2	
3	3	F		×	3 3 2 F	3 3 2 F	3
F	3	2	○	×	×	×	×

$\delta(q_1, \mathrm{a})\delta(q_2, \mathrm{a})$
$\delta(q_1, \mathrm{b})\delta(q_2, \mathrm{b})$

状態併合表を用いた最小化

Q	遷移先 a	遷移先 b	受理	0	1	2	3
0	1			0			
1	3	2		×	1		
2	3	2		×	~~3 3~~ ~~2 2~~	2	
3	3	F		×	~~3 3~~ 2 F	~~3 3~~ 2 F	3
F	3	2	○	×	×	×	×

同一遷移先なら
判定不要

状態併合表を用いた最小化

Q	遷移先		受理				
	a	b					
0	1			0			
1	3	2		×	1		
2	3	2		×		2	
3	3	F		×	×2 F	×2 F	3
F	3	2	○	×	×	×	×

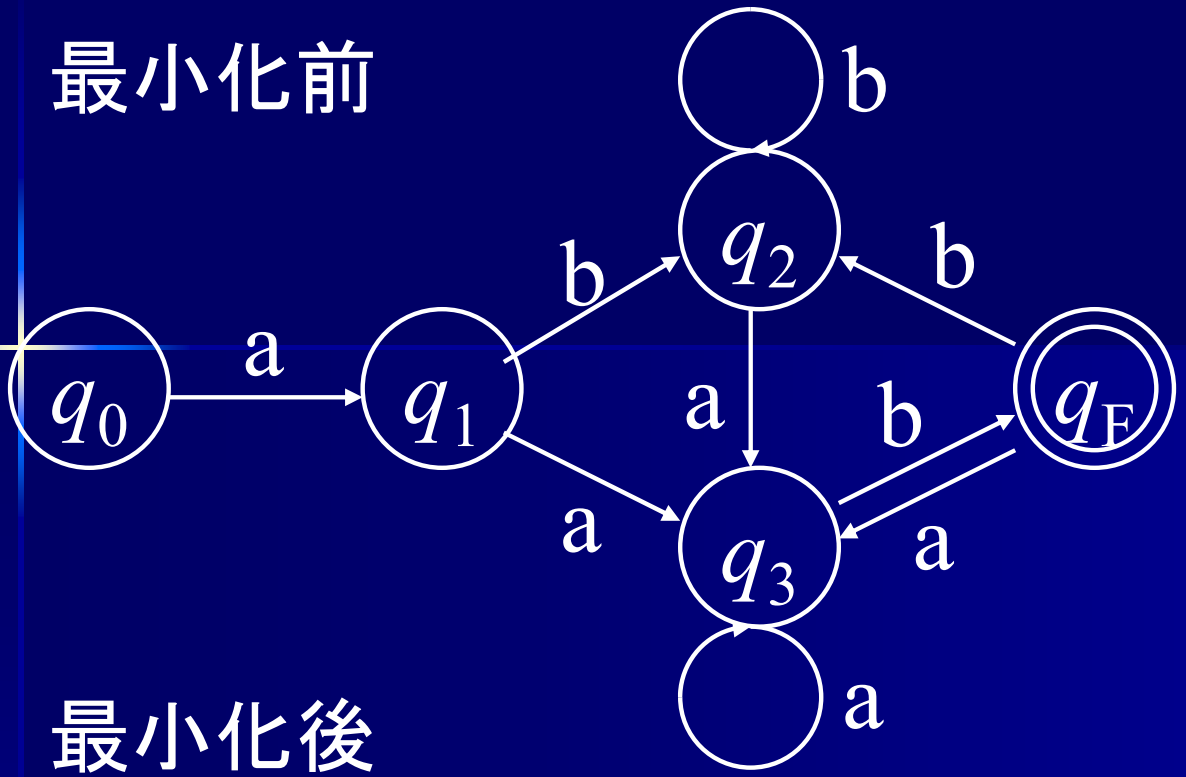
q_2q_F は×なので
2 F に×を付ける

状態併合表を用いた最小化

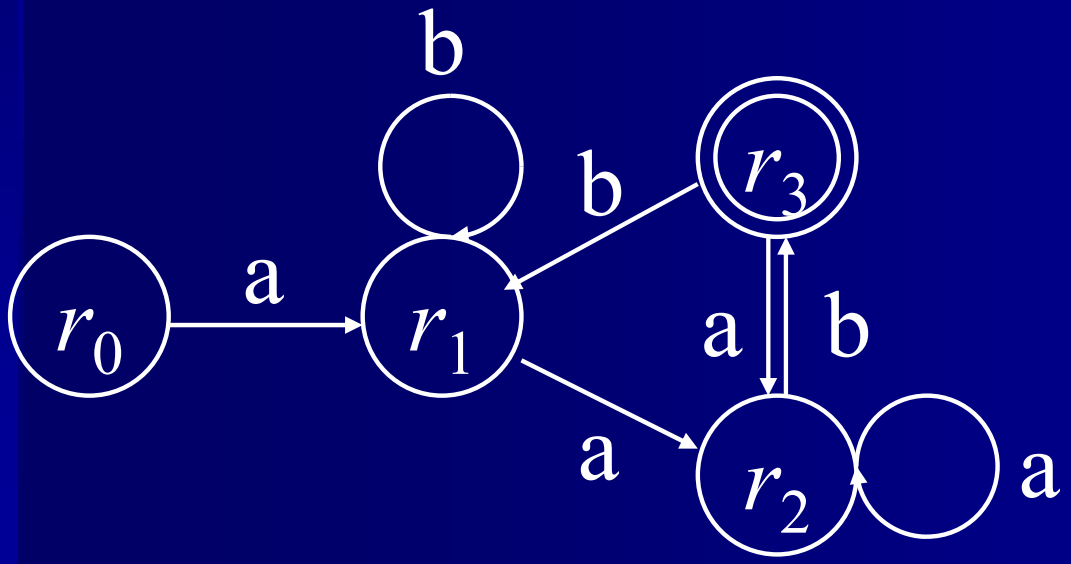
Q	遷移先		受理				
	a	b					
0	1			0			
1	3	2		×	1		
2	3	2		×		2	
3	3	F		×	×2 F	×2 F	3
F	3	2	○	×	×	×	×

最後まで×が付かなかった $\{q_1, q_2\}$ が等価

最小化前



最小化後

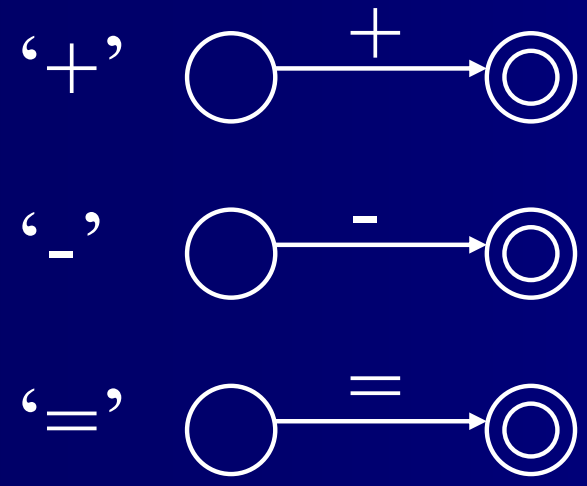


決定性有限オートマトンの作成
情報システムプロジェクトIの場合

- マイクロ構文

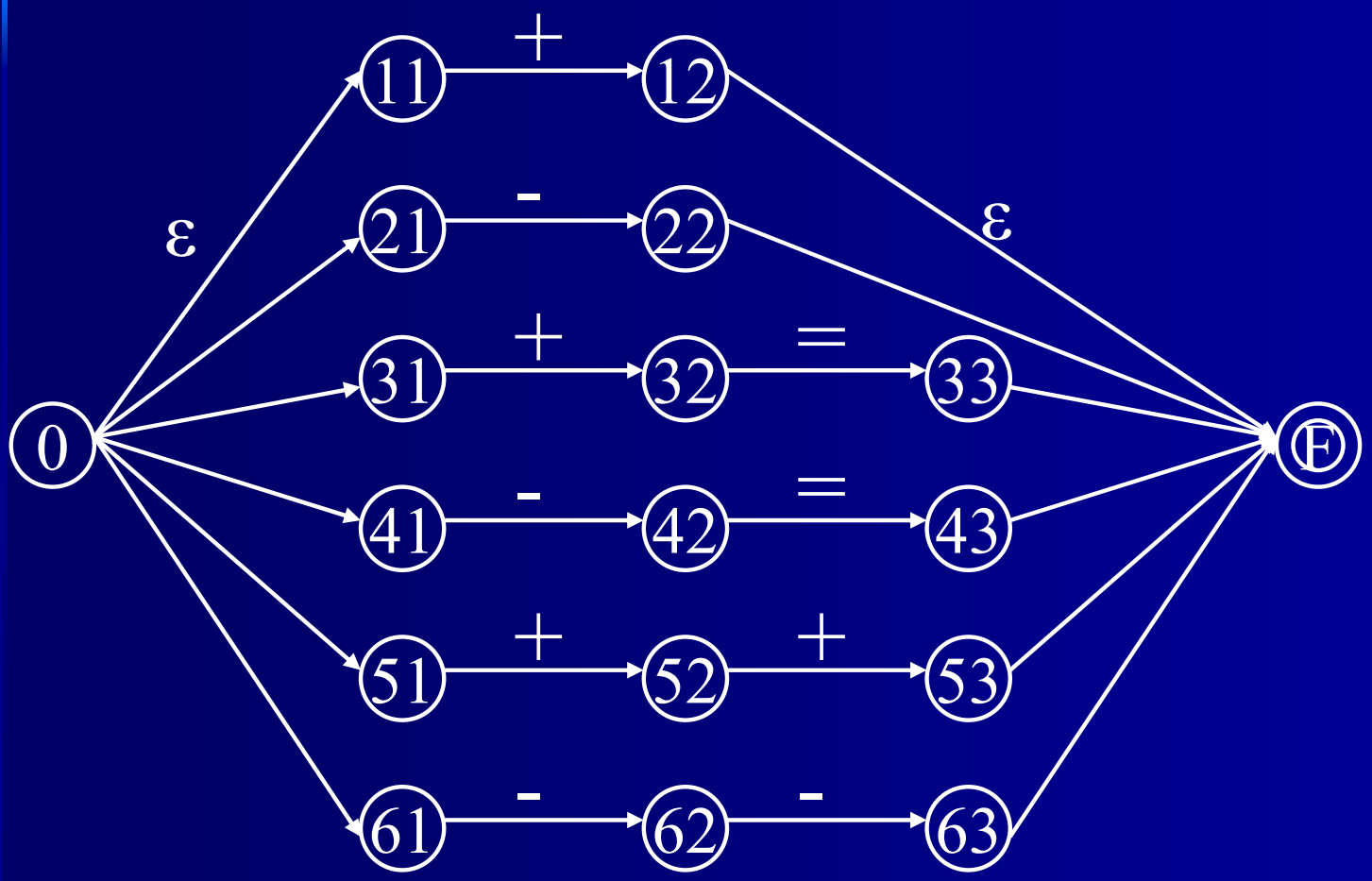
OPERATOR(一部) ::=

‘+’ | ‘-’ | ‘+’ ‘=’ | ‘-’ ‘=’ | ‘+’ ‘+’ | ‘-’ ‘-’



非決定性オートマトンへ

‘+’ | ‘-’ | ‘+’ ‘=’ | ‘-’ ‘=’ | ‘+’ ‘+’ | ‘-’ ‘-’



決定性オートマトンへ

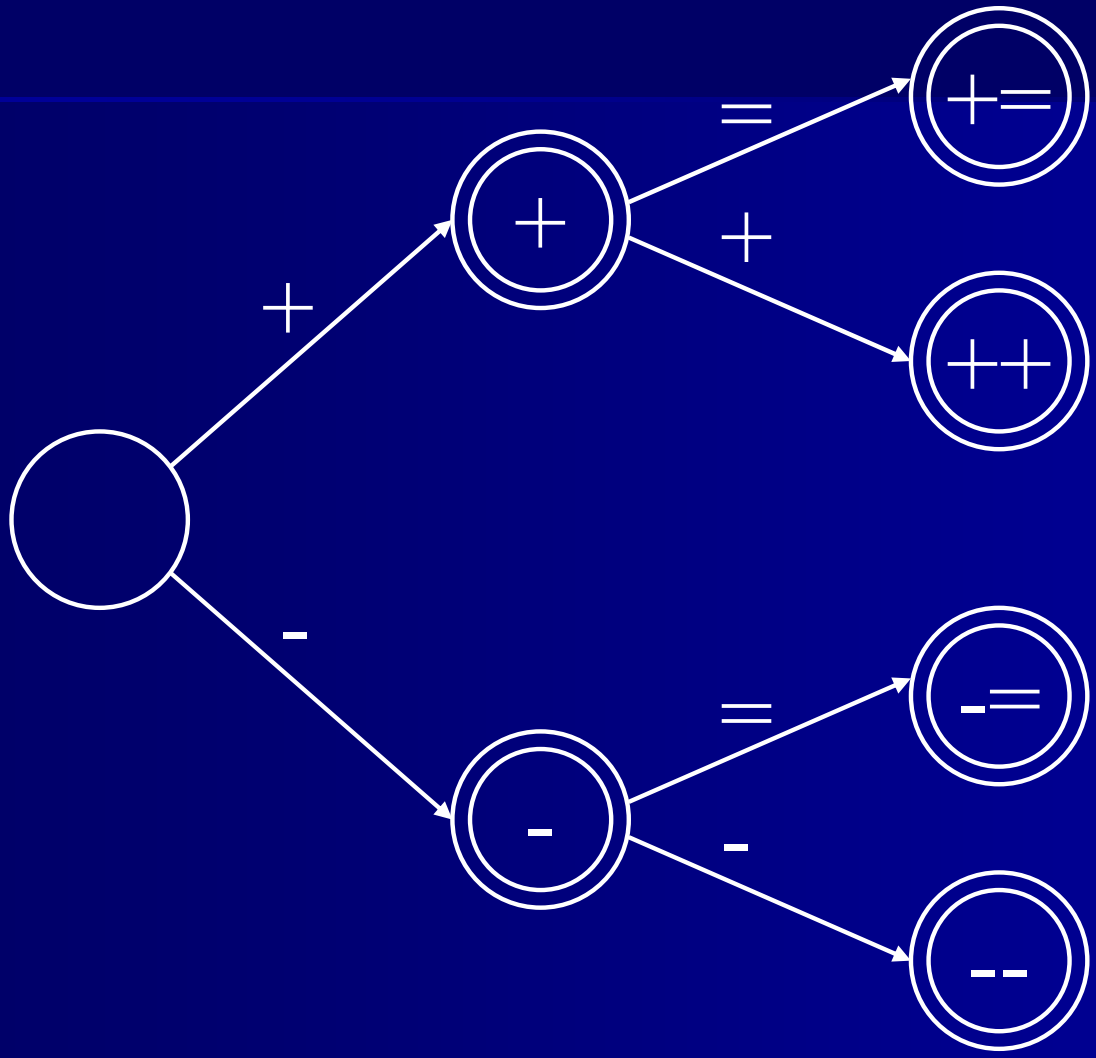
Q_N	ε	+	-	=
0	0,11,21,31, 41,51,61			
11	11	12,F		
12	12,F			
21	21		22,F	
22	22,F			
31	31	32		
32	32			33,F
33	33,F			

Q_N	ε	+	-	=
41	41		42	
42	42			43,F
43	43,F			
51	51	52		
52	52	53,F		
53	53,F			
61	61		62	
62	62		63,F	
63	63,F			

決定性オートマトンへ

Q_D	ε	+	-	=
0,11,21,31, 41,51,61	0,11,21,31, 41,51,61	12,32, 52,F	22,42, 62,F	
12,32,52,F	12,32,52,F	53,F		33,F
22,42,62,F	22,42,62,F		63,F	43,F
33,F	33,F			
43,F	43,F			
53,F	53,F			
63,F	63,F			

決定性オートマトン



状態数最小化は不要

決定性有限オートマトン(一部)