情報技術史論

計算機の機構と計算理論
有限オートマトン

担当：情報理工学専攻　知能ソフトウェア研究室

復習：計算可能性の概念

3つの重要な計算モデル（1930年代）

- **チューリング機械（チューリング）**
- ラムダ計算（チャーチ、ロッサー）
- **帰納関数（エルブラン、ゲーデル、クリーネ）**

状態遷移に基づくモデル
（**オートマトン**）

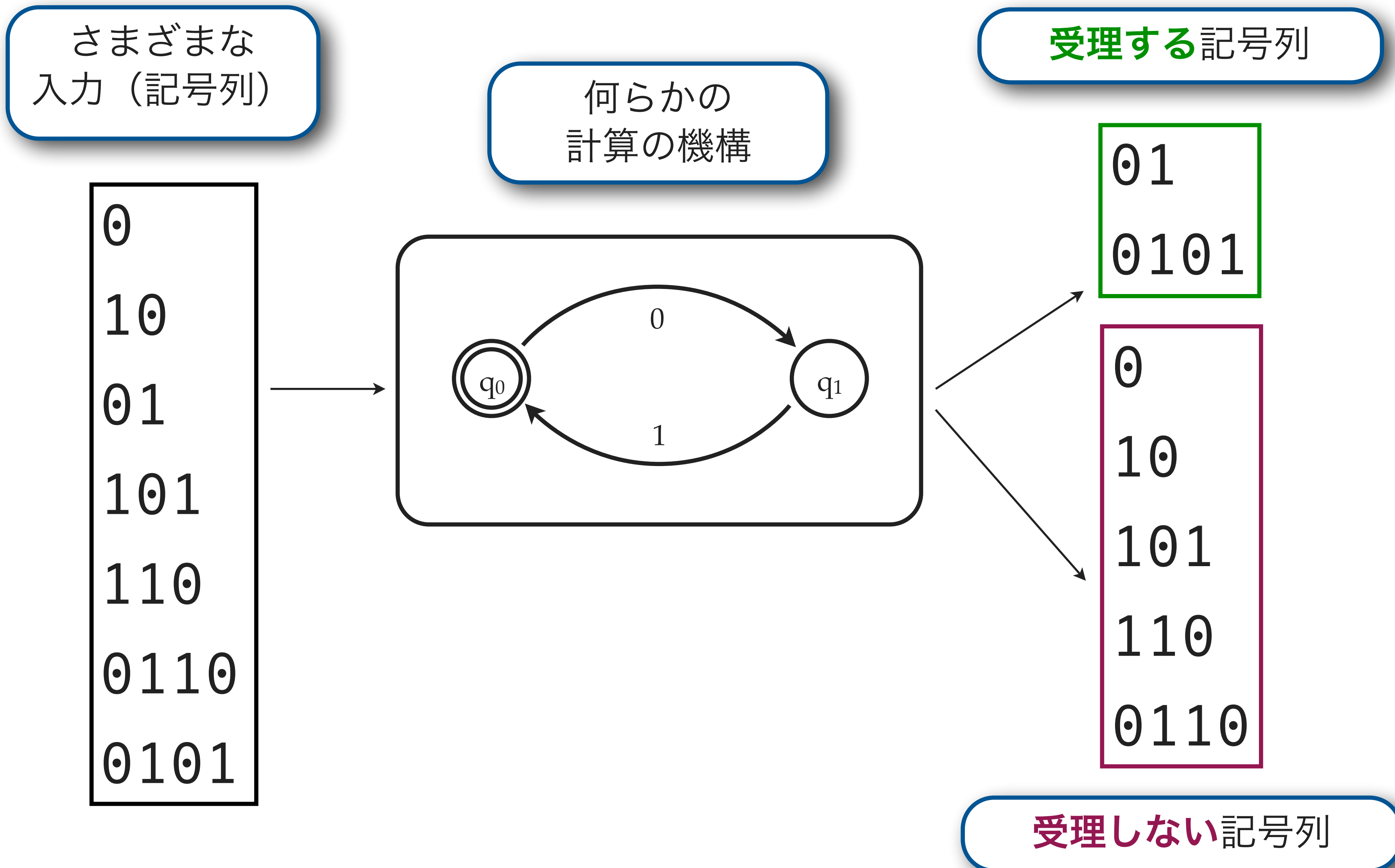
定理（上記計算モデルは等価であり、十分な能力を持つ）

これらの3つの計算モデルの能力は等しい (互いに他をシミュレート可能)。
また、実際のコンピュータの能力とも等しい。

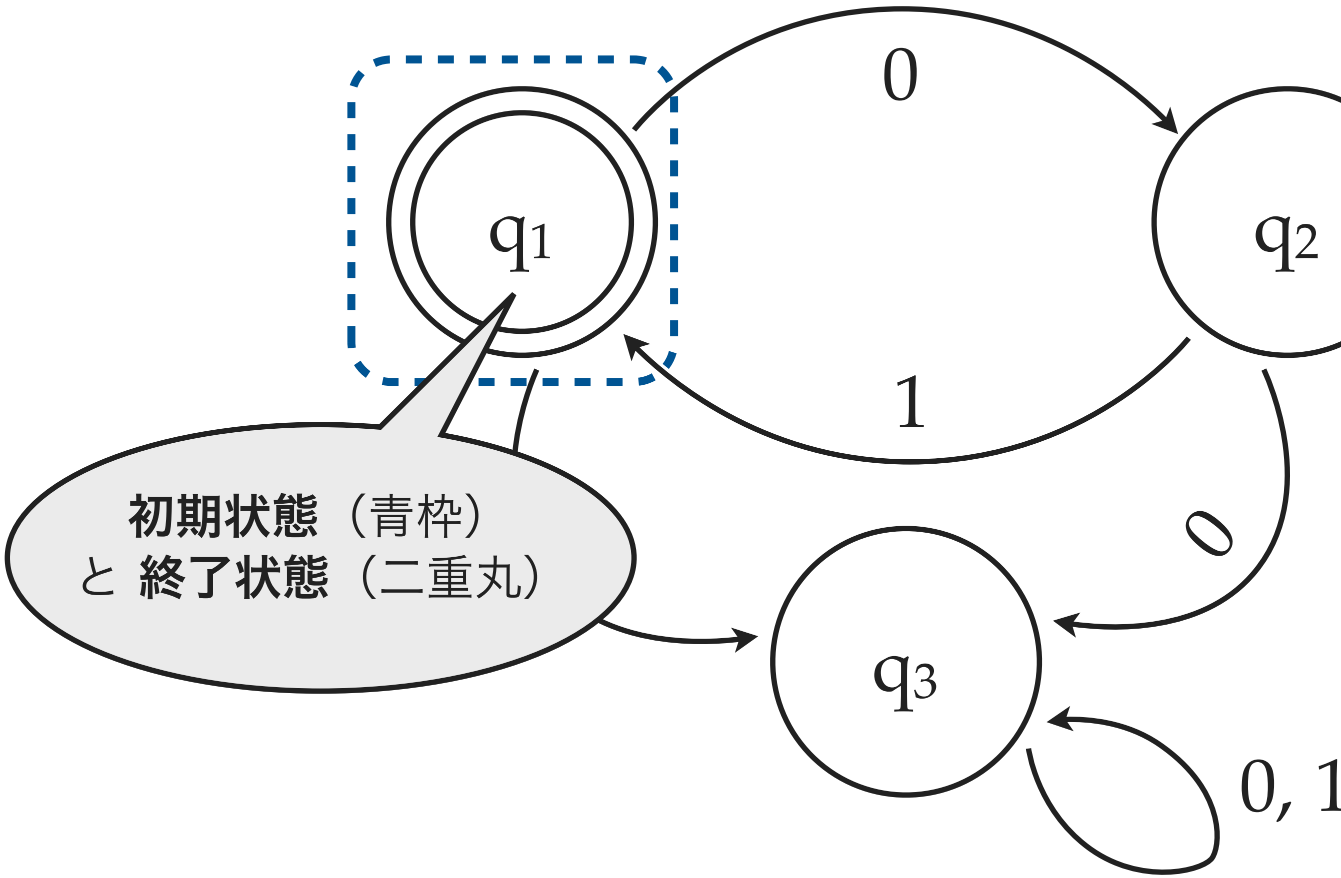
チャーチの提唱

これらの計算モデルでできることを「計算可能である」ことの定義とする

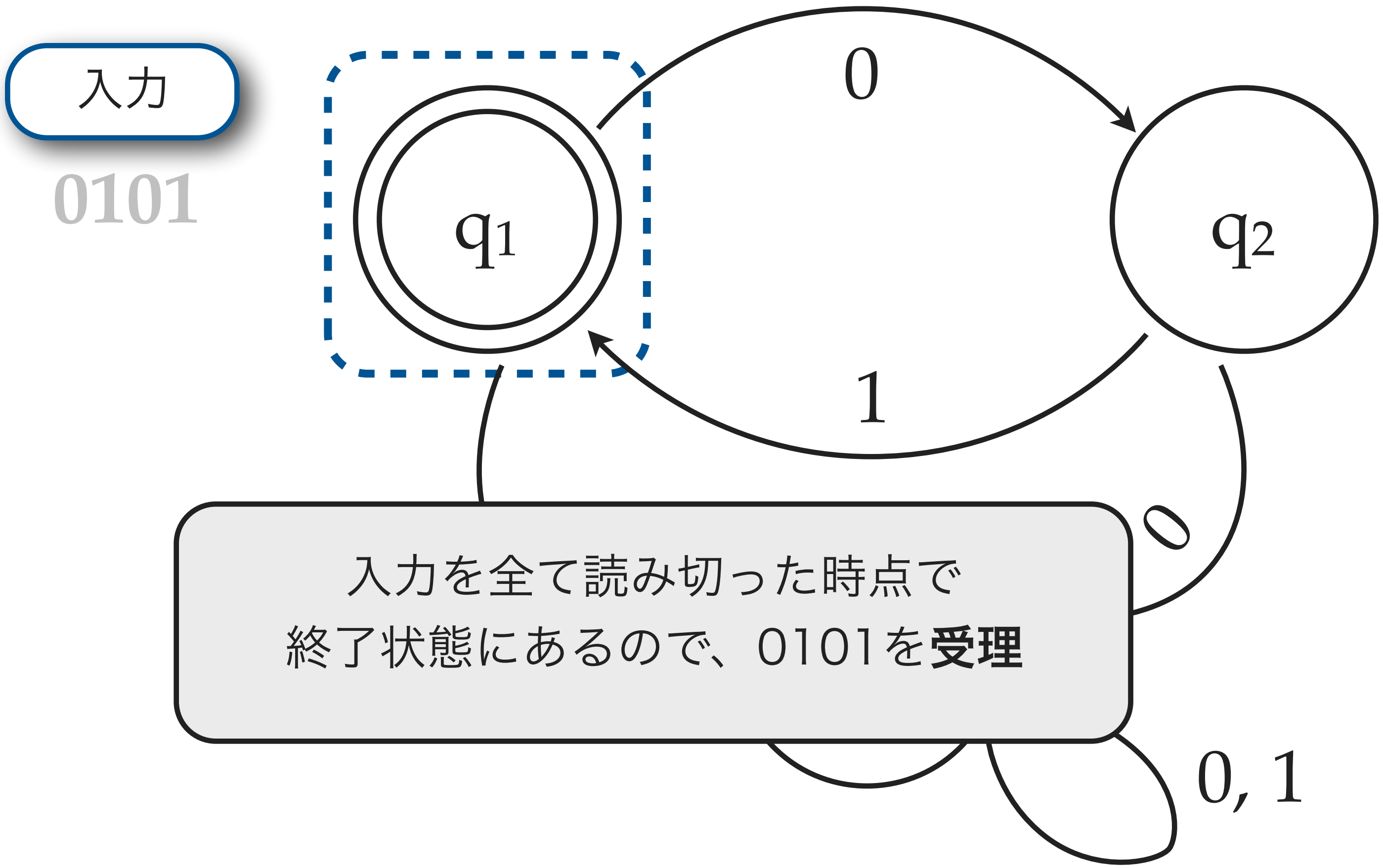
計算：入力される記号列を2種類（受理・非受理）に分類



例：入力を0,1の列とし、「01」の0回以上の繰り返しを受理する



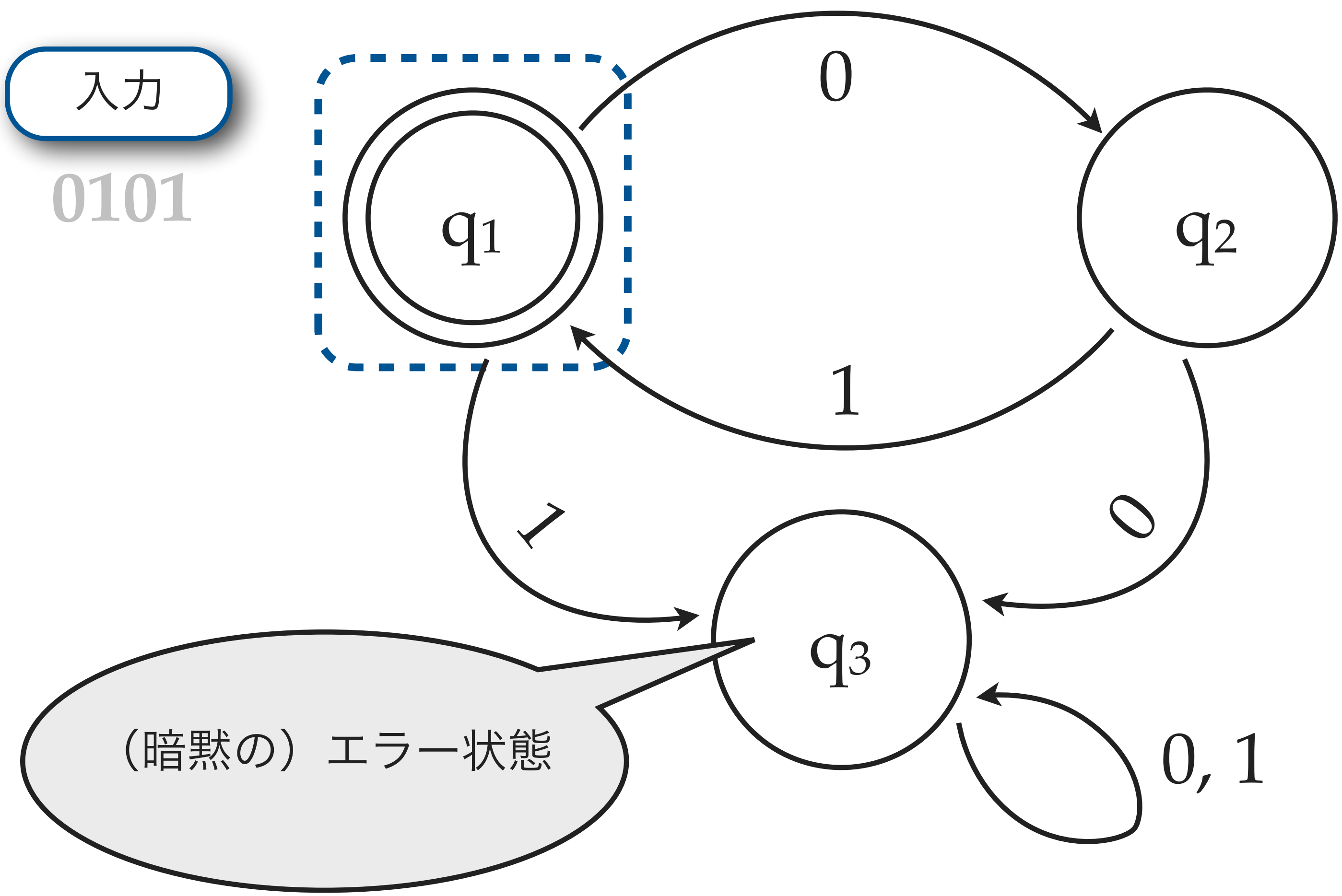
例：入力を0,1の列とし、「01」の0回以上の繰り返しを受理する



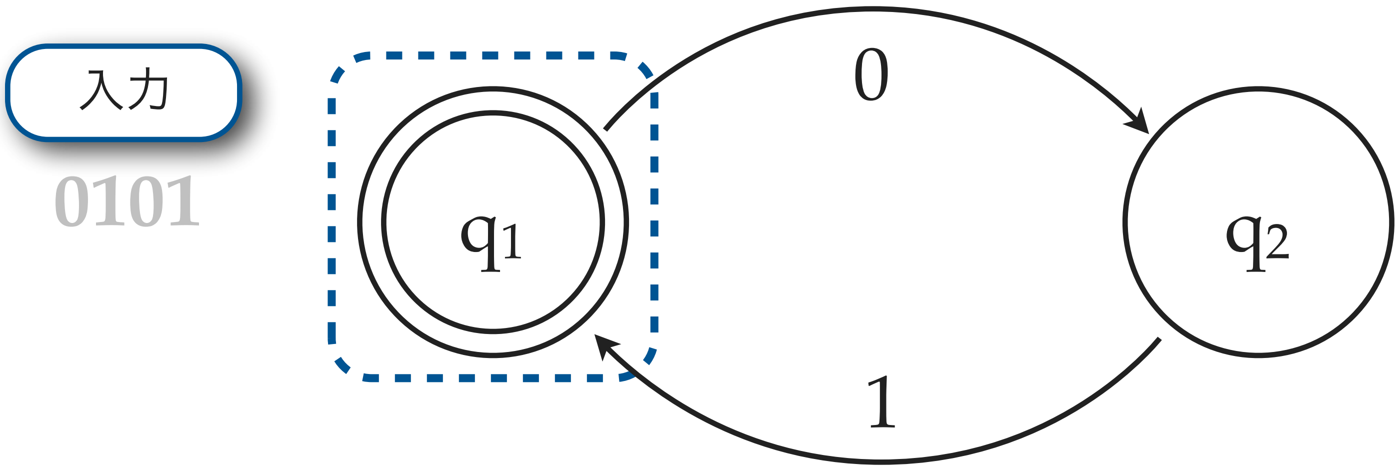
入力を全て読み切った時点で
終了状態にあるので、0101を**受理**

有限オートマトン（Finite Automata, FA, DFA）

例：入力を0,1の列とし、「01」の0回以上の繰り返しを受理する



例：入力を0,1の列とし、「01」の0回以上の繰り返しを受理する



明示されていない遷移は**エラー**
（非受理）とする

有限オートマトン（Finite Automata, FA, DFA）

定義：有限オートマトン

有限オートマトンとは、以下の要素からなる5つ組（Q, Σ, δ, q_1, F）である。

・状態の有限集合 Q
・入力記号の有限集合 （アルファベット）Σ
・遷移関数 δ：Q × Σ → Q
・初期状態 $q_1 \in Q$
・終了状態の集合 $F \subseteq Q$

- 遷移先は一意（**決定的**）
- 初期状態はただ一つ
- 終了状態は複数あってよい

FAの実装例

```
s = q1
i = 0
while (i < n) {
   s = δ[s][input[i]]
   i++
}
return F[s]
```

例：後置記法の数式 a b + c × … を受理

数字を0, 1とし、1ビット演算 (1+1=0) に限定

受理する数式

0
1
0 1 +
1 0 ×
0 0 + 1 ×
1 1 × 1 + 1 ×

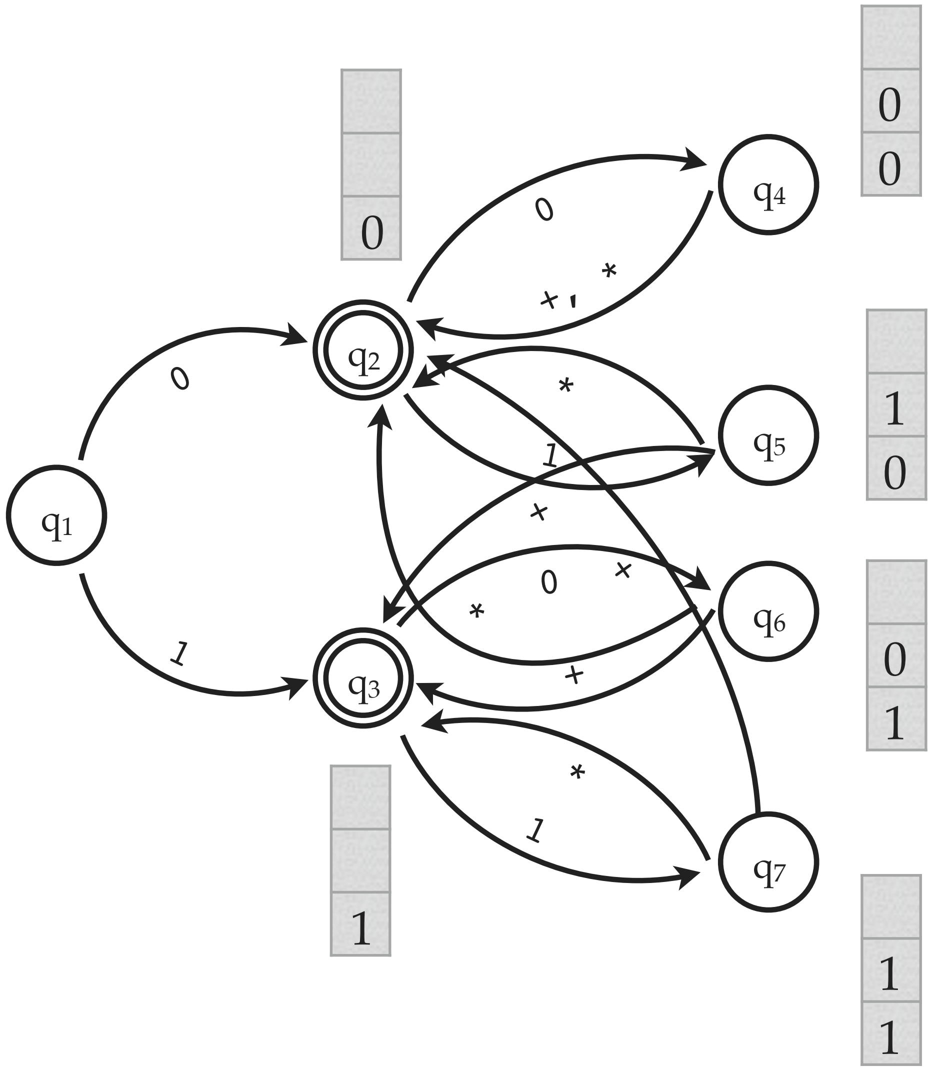
受理しない数式

ε　空列
+
0 ×
0 1 0
1 0 × +
1 1 × 1 + 1 0

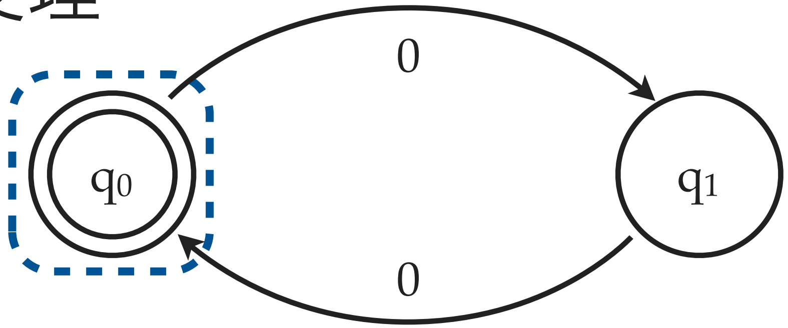
Σ = {0,1,+,*}　（*は乗算）

Q = {q_1,...}

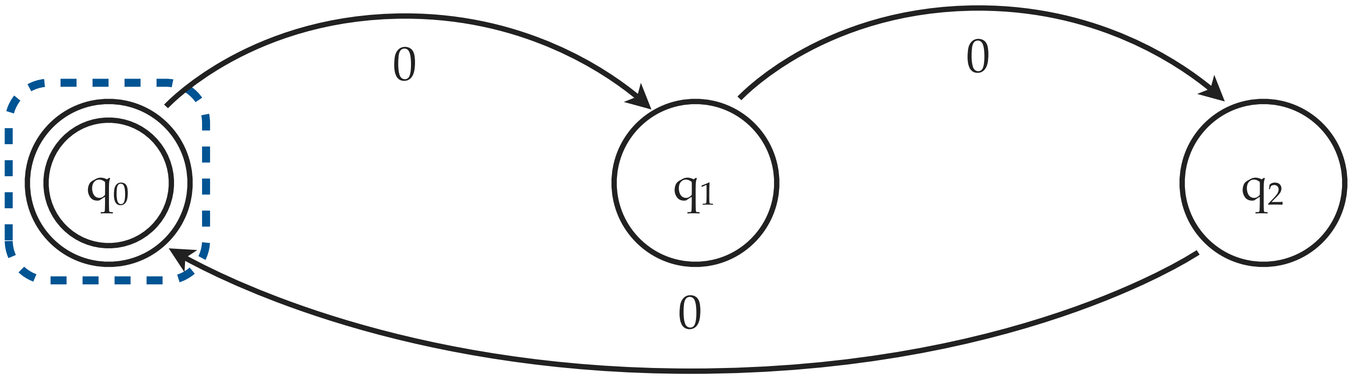
初期状態 = q_1



例：偶数個の0を受理



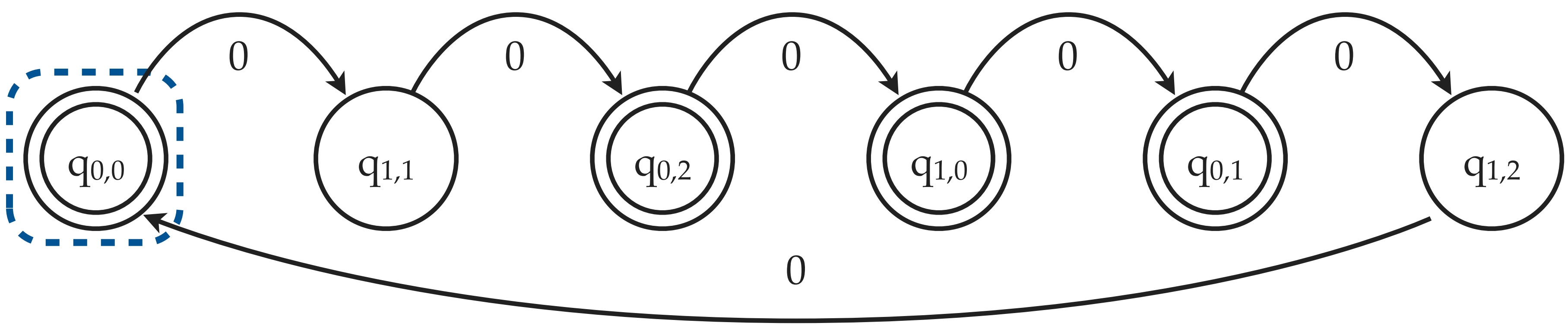
例：3の倍数個の0を受理



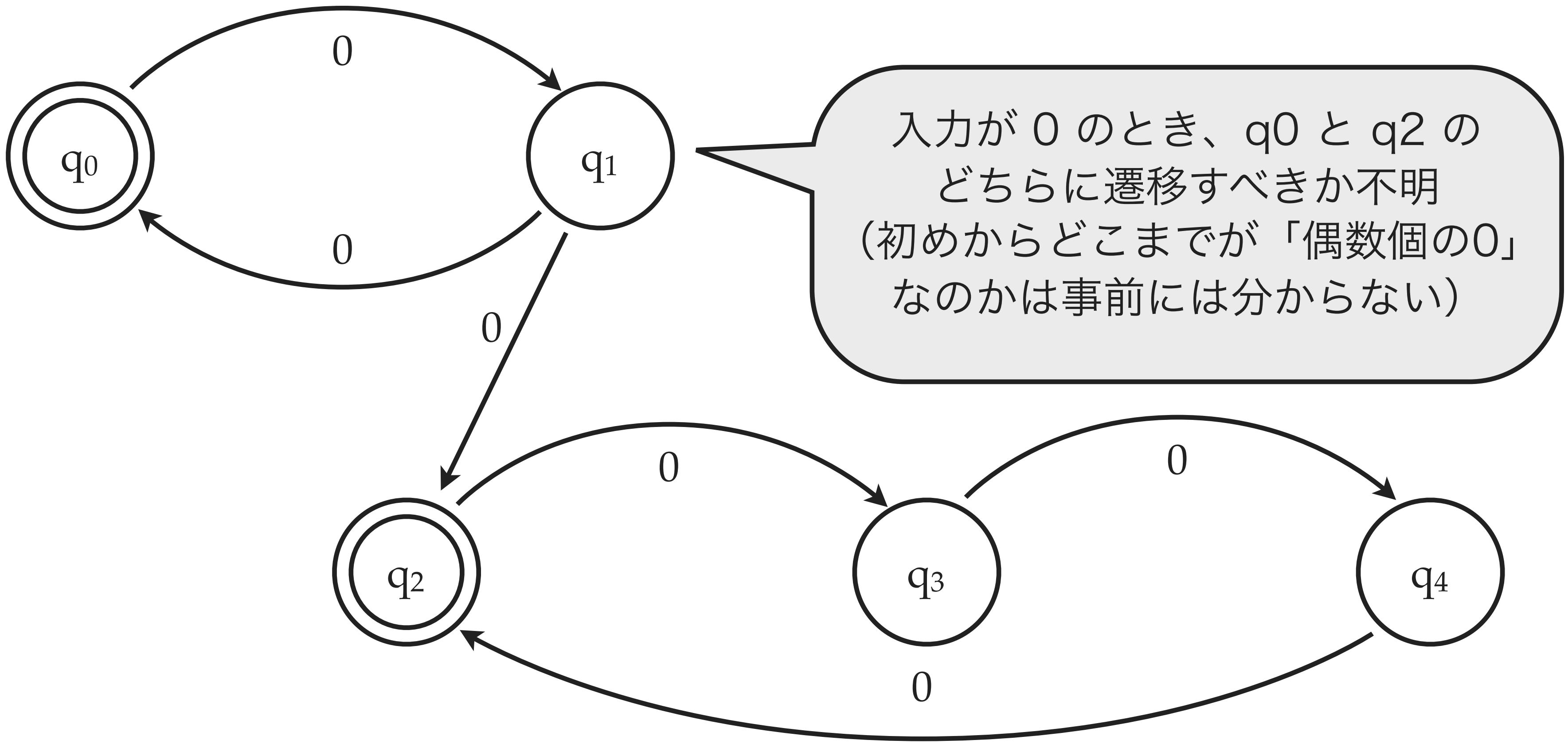
例：偶数個の0 **または** 3の倍数個の0を受理

読み込んだ0の個数 x の
「x mod 2（2で割った余り）」と
「x mod 3」の**組み合わせ**を
状態とする

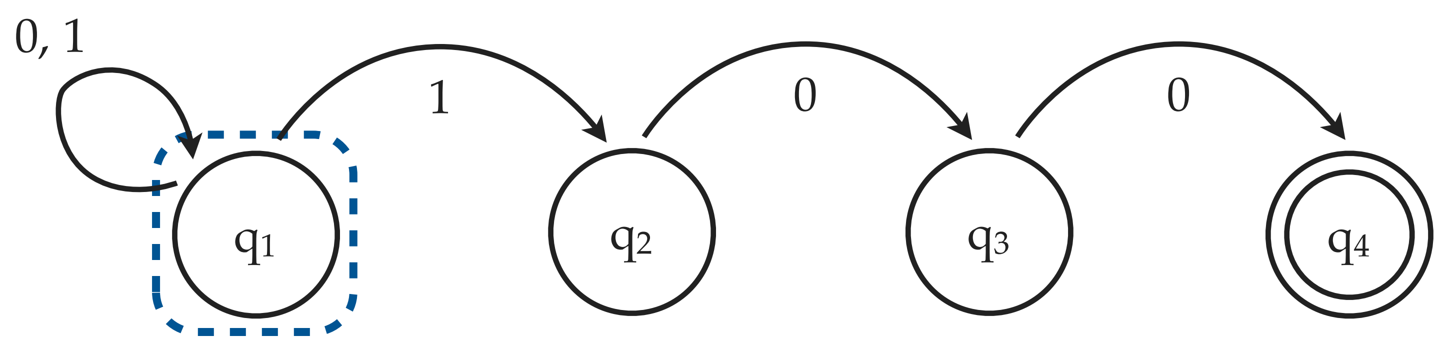
	mod3 = 0	mod3 = 1	mod3 = 2
mod2 = 0	$q_{0,0}$	$q_{0,1}$	$q_{0,2}$
mod2 = 1	$q_{1,0}$	$q_{1,1}$	$q_{1,2}$



例：偶数個の0**の後に**、3の倍数個の0が続く記号列



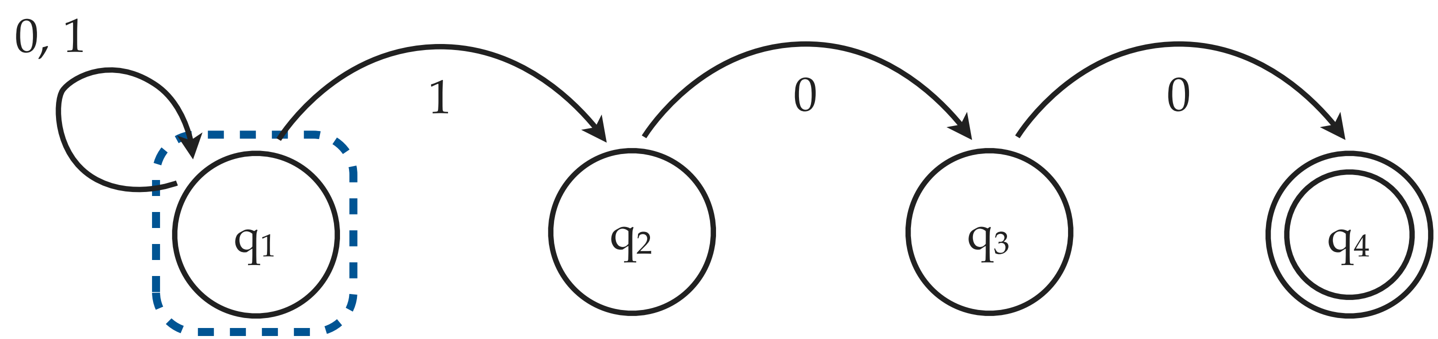
例：0,1からなり、末尾が100である記号列を受理



非決定性有限オートマトン (NFA)

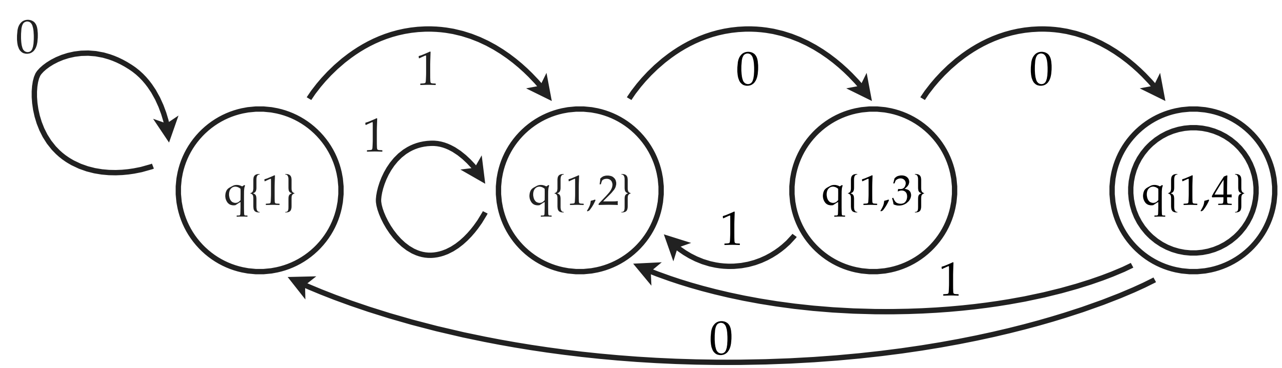
- 可能な遷移先が複数あってよい（**非決定性**）
  - 遷移関数 $\delta : Q \times \Sigma \rightarrow P(Q)$ (Qのべき集合)
- 入力を全て読み終了状態に到達する遷移が**存在すれば**受理
- 「入力を読まない遷移」があってもよい
  - ε を「空列」を意味する記号とし、アルファベットに含める

例：0,1からなり、末尾が100である記号列を受理



定理（FAとNFAの等価性）

NFAは、同じ記号列を受理するFA（決定性オートマトン）に変換可能



有限オートマトンで受理できない例

例：aが0個以上続き、それと同数のbが続く列を受理

受理する列　ε, ab, aabb, aaabbb, ...

一番初めに現れる b より前に、a がいくつ現れたかを**記憶**する必要がある。しかし a の数に制限はないので、**有限**の状態では実現不能

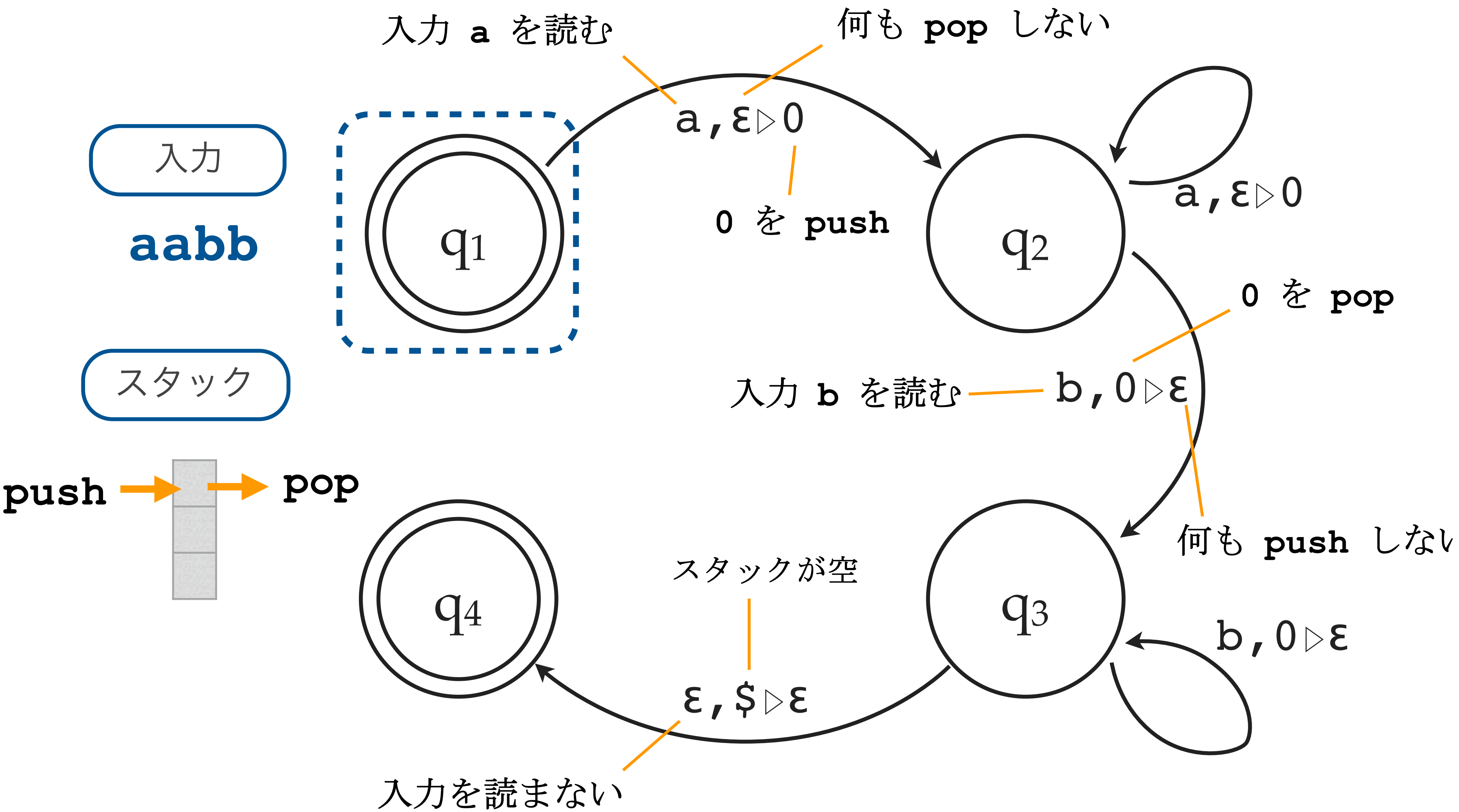
「現在の状態」に加え、**記憶装置**を持ったオートマトンを考える

スタックを使う：**プッシュダウンオートマトン**

テープを使う：**チューリング機械**

スタック：後入れ先出し(Last In First Out)の制約のあるデータ構造
テープ：制約のないデータ構造

例：aが0個以上続き、それと同数のbが続く列を受理



（非決定性）プッシュダウンオートマトン(PDA)

定義：プッシュダウンオートマトン

プッシュダウンオートマトンとは、以下の要素からなる6つ組
$<Q, \Sigma, \Gamma, \delta, q_1, F>$ である。

- 状態の有限集合 Q
- アルファベット Σ
- **スタックに格納されるアルファベット Γ**
- 遷移関数 $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow P(Q \times \Gamma^*)$　　（$\Sigma_\varepsilon = \Sigma \cup \{ \varepsilon \}$）
- 初期状態 $q_1 \in Q$
- 終了状態の集合 $F \subseteq Q$

スタックに追加する
アルファベットの**列**

PDAで受理できない集合の例

例：aが0個以上続き、それと同数のbが続き、
さらにそれと同数のcが続く列を受理

受理する列　ε, abc, aabbcc, aaabbbccc, ...

スタックを2つ持つPDAなら、この例も受理可能になる。

それだけでなく、スタックをそれ以上追加しても能力は変わらない

（スタックを2つ持つPDAは、チューリング機械と等価な能力を持つ）

参考文献

- 「オートマトン 言語理論 計算論 I (II)」（第2版）
  J. Hopcroft, R. Motwani, J. Ullman 著，野崎 昭弘ら 訳，サイエンス社
- 「計算理論の基礎 1. オートマトンと言語」「計算理論の基礎 2. 計算可能性の理論」
  Michael Sipser 著，太田 和夫 監訳，阿部 正幸 訳，共立出版
- 「計算モデル論入門」井田 哲雄・浜名 誠 著，サイエンス社