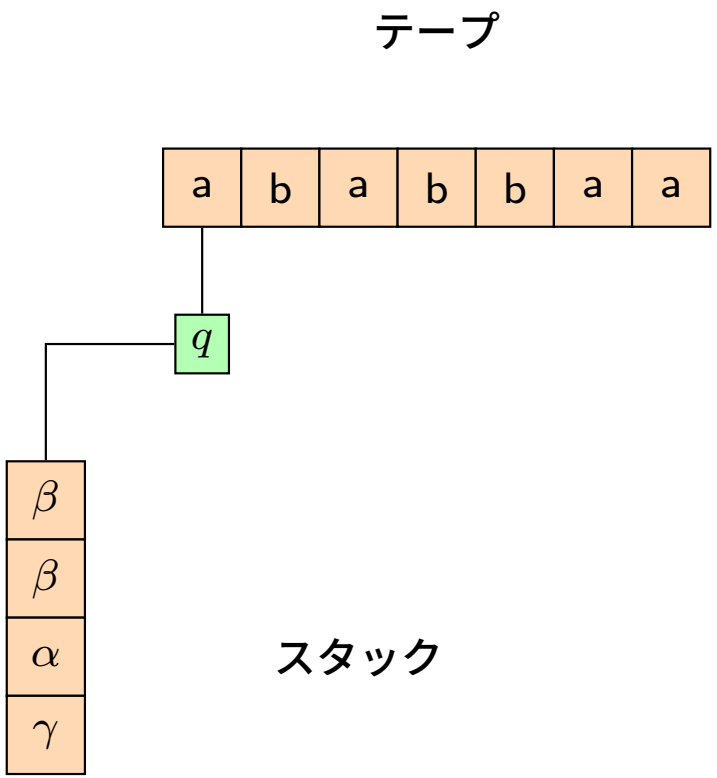
プッシュダウンオートマトン

離散数学・オートマトン
2024 年後期
佐賀大学理工学部　只木進一

1. プッシュダウンオートマトン: Pushdown automata
2. 決定性プッシュダウンオートマトン: Deterministic PDA
3. PDA と受理言語: PDA and Their Accepted Languages
4. 非決定性プッシュダウンオートマトン: Nondeterministic PDA

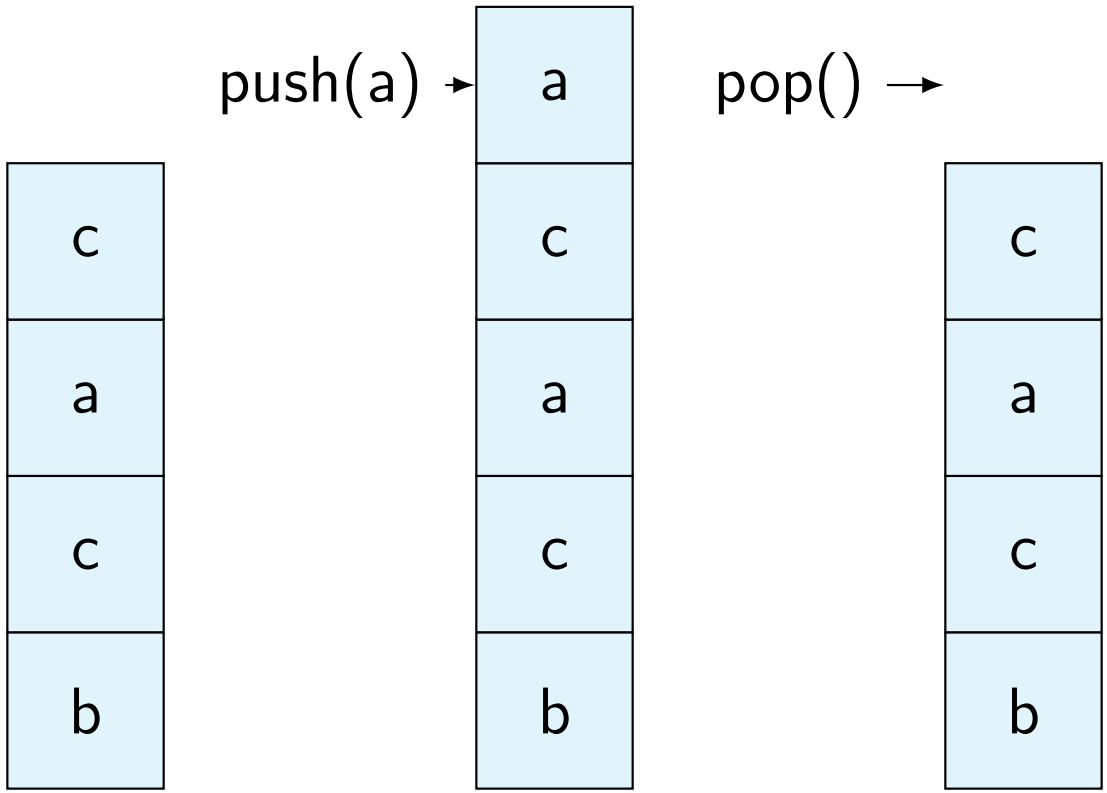
プッシュダウンオートマトン: 動作イメージ

- テープとともに、スタックの文字を読む
- 状態遷移するとともに、スタックへ文字列を書き込む
- スタックという特殊な無限に大きなメモリを持つ機械



スタック: stacks

- リストのような 1 次元のデータ列
- 先頭に書く (push) ことと、先頭から読む (pop) ことだけが許される
  - 先頭以外のデータは触れない
  - FILO (First-In Last-Out)
  - pop: 先頭を取り出して読む、つまり、先頭の要素はスタックから無くなることに注意
- Python での実装例: 次シート
  - `deque` を利用
  - `append()`: 最後に要素を追加
  - `pop()`: 最後の要素を取り出し、削除


push(a)
a
c
a
c
b
pop()
c
a
c
b
c
a
c
b

Stack クラス定義

```python
from collections import deque
from typing import TypeVar, Generic
T = TypeVar('T')
class Stack(Generic[T]):
    """
    スタックのクラス
    """
    def __init__(self):#コンストラクタ
        self.elements:deque[T] = deque()
    def is_empty(self) -> bool:#要素が無いとき True
        return len(self.elements) == 0
    def push(self,e:T) -> None:#要素を追加
        self.elements.append(e)
    def pop(self) -> T:#末尾の要素を取り出す。要素は削除される
        return self.elements.pop()
    def peek(self) -> T:#末尾の要素を調べる
        return self.elements[-1]
    def size(self) -> int:#要素数を返す
        return len(self.elements)
    def __str__(self) -> str:#文字列化
        s = '['
        for x in self.elements:
            s += str(x)+','
        s = s.removesuffix(',')
        s += ']'
        return s
```

Stack クラス利用例

```
1  myStack = Stack[str]()
2  myStack.push('a')
3  myStack.push('b')
4  myStack.push('b')
5  print(myStack)
6
7  myStack.pop()
8  myStack.pop()
9  print(myStack)
```

https://github.com/discrete-math-saga/PDA

決定性プッシュダウンオートマトン
Deterministic Pushdown Automata

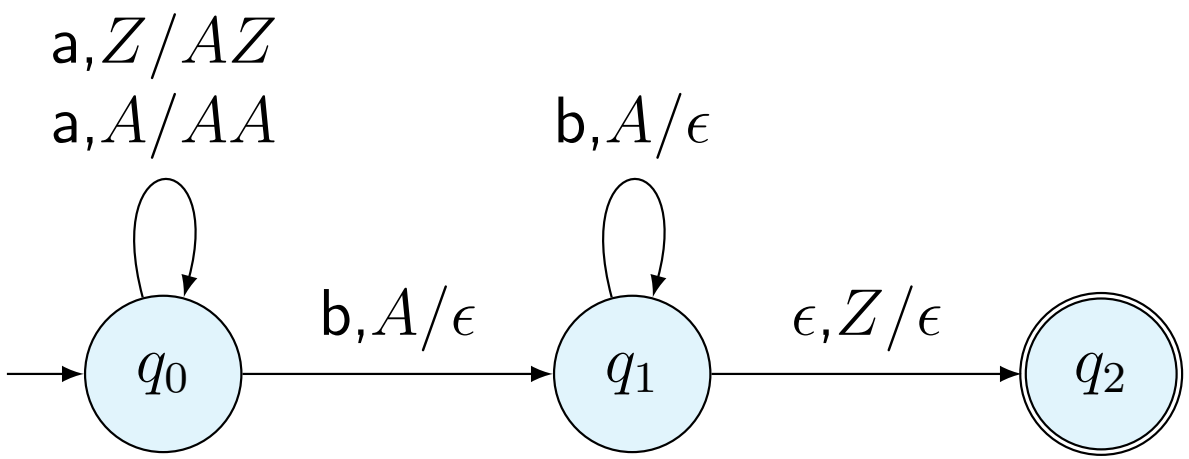
$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle \tag{2.1}$$

- Q: 内部状態の集合
- Σ: テープのアルファベット
- Γ: スタックのアルファベット
- $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$: 遷移関数
  - 注意: スタックから 1 文字読み、文字列を書き込む
  - 文字列は、右側からスタックに書き込む
- $q_0 \in Q$: 初期状態
- $Z_0 \in \Gamma$: スタックの底の記号
- $F \subseteq Q$: 終状態の集合

例 2.1:

$$Q = \{q_0, q_1, q_2\}, \qquad F = \{q_2\},$$
$$\Sigma = \{\mathsf{a}, \mathsf{b}\}, \qquad \Gamma = \{A, Z\}.$$

$$\delta(q_0, \mathsf{a}, Z) = (q_0, AZ), \quad \delta(q_0, \mathsf{a}, A) = (q_0, A), \quad \delta(q_0, \mathsf{b}, A) = (q_0, \epsilon),$$
$$\delta(q_1, \mathsf{b}, A) = (q_1, \epsilon), \qquad \delta(q_1, \epsilon, Z) = (q_2, \epsilon).$$



注意:$\delta(q_1, \epsilon, Z)$ は決定的動作であることに注意。スタック文字が Z のときのみ。

動作: (Q, Σ^*, Γ^*) の変化

$(q_0, \mathsf{aaabbb}, Z) \vdash (q_0, \mathsf{aabbb}, AZ)$ aを読んでいる間はq_0に留まる

$\vdash (q_0, \mathsf{abbb}, AAZ)$

$\vdash (q_0, \mathsf{bbb}, AAAZ)$ bを読むとq_1へ遷移

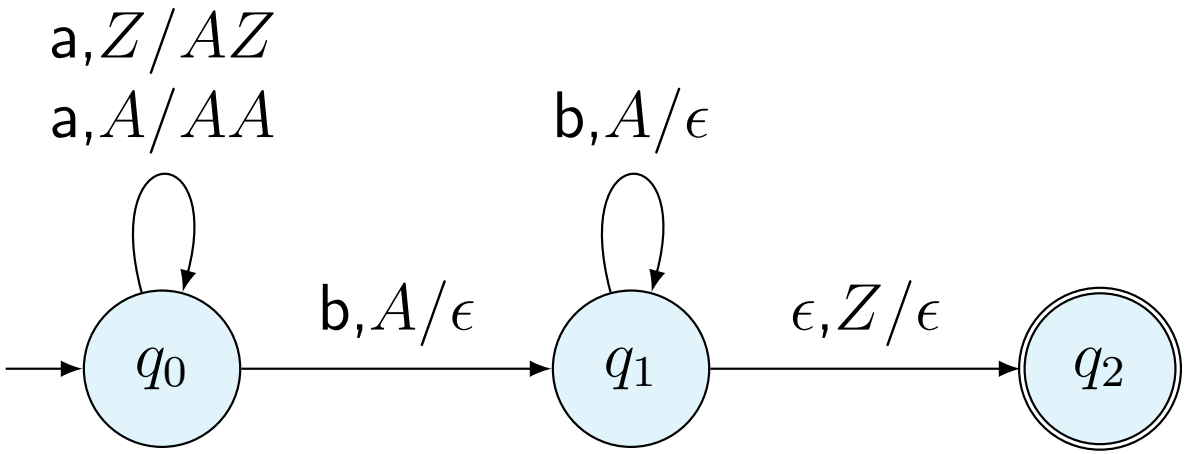
$\vdash (q_1, \mathsf{bb}, AAZ)$

$\vdash (q_1, \mathsf{b}, AZ)$

$\vdash (q_1, \epsilon, Z)$ 空スタックでq_2へ遷移

$\vdash (q_2, \epsilon, \epsilon)$

aabb に対する動作



動作失敗: a と b の数が異なる

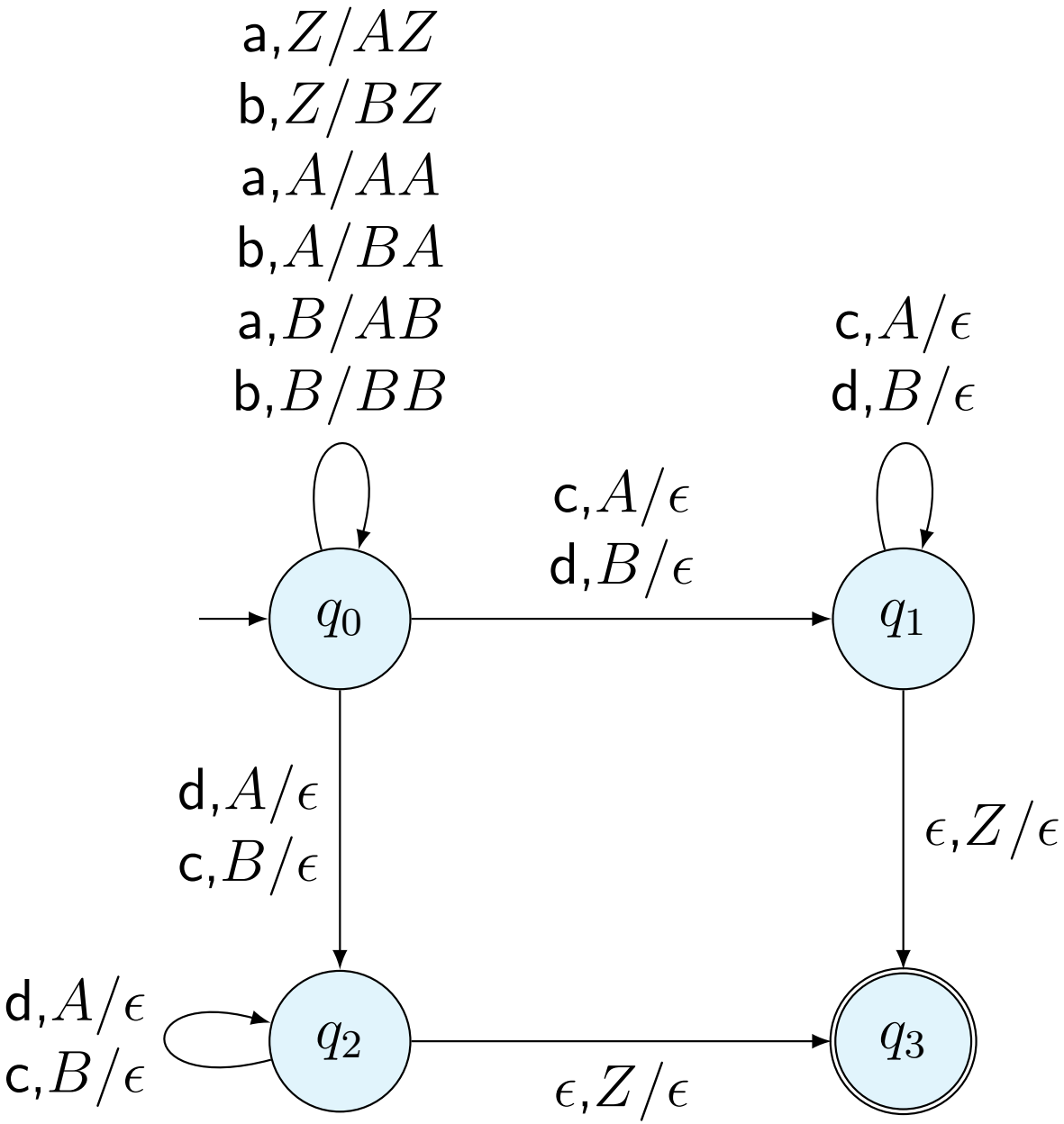
$$
\begin{aligned}
(q_0, \mathsf{aaabb}, X) &\vdash (q_0, \mathsf{aabb}, AZ) \\
&\vdash (q_0, \mathsf{abb}, AAZ) \\
&\vdash (q_0, \mathsf{bb}, AAAZ) \\
&\vdash (q_1, \mathsf{b}, AAZ) \\
&\vdash (q_1, \epsilon, AZ)
\end{aligned}
$$

例 2.2:

$$Q = \{q_0, q_1, q_2, q_3\}, \qquad F = \{q_3\},$$
$$\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}, \qquad \Gamma = \{A, B, Z\}.$$

$$\delta(q_0, \mathsf{a}, Z) = (q_0, AZ), \qquad \delta(q_0, \mathsf{b}, Z) = (q_0, BZ),$$
$$\delta(q_0, \mathsf{a}, A) = (q_0, AA), \qquad \delta(q_0, \mathsf{b}, A) = (q_0, BA),$$
$$\delta(q_0, \mathsf{a}, B) = (q_0, AB), \qquad \delta(q_0, \mathsf{b}, B) = (q_0, BB),$$
$$\delta(q_0, \mathsf{c}, A) = (q_1, \epsilon), \qquad \delta(q_0, \mathsf{d}, B) = (q_1, \epsilon),$$
$$\delta(q_0, \mathsf{d}, A) = (q_2, \epsilon), \qquad \delta(q_0, \mathsf{c}, B) = (q_2, \epsilon),$$
$$\delta(q_1, \mathsf{c}, A) = (q_1, \epsilon), \qquad \delta(q_1, \mathsf{d}, B) = (q_1, \epsilon),$$
$$\delta(q_2, \mathsf{d}, A) = (q_2, \epsilon), \qquad \delta(q_2, \mathsf{c}, B) = (q_2, \epsilon),$$
$$\delta(q_1, \epsilon, Z) = (q_3, \epsilon), \qquad \delta(q_2, \epsilon, Z) = (q_3, \epsilon)$$

注意:$\delta(q_1, \epsilon, Z)$ は決定的動作であることに注意。スタック文字が Z のときのみ。


a, Z/AZ
b, Z/BZ
a, A/AA
b, A/BA
a, B/AB
b, B/BB
c, A/ϵ
d, B/ϵ
c, A/ϵ
d, B/ϵ
q_0
q_1
d, A/ϵ
c, B/ϵ
$\epsilon, Z/\epsilon$
d, A/ϵ
c, B/ϵ
q_2
q_3
$\epsilon, Z/\epsilon$

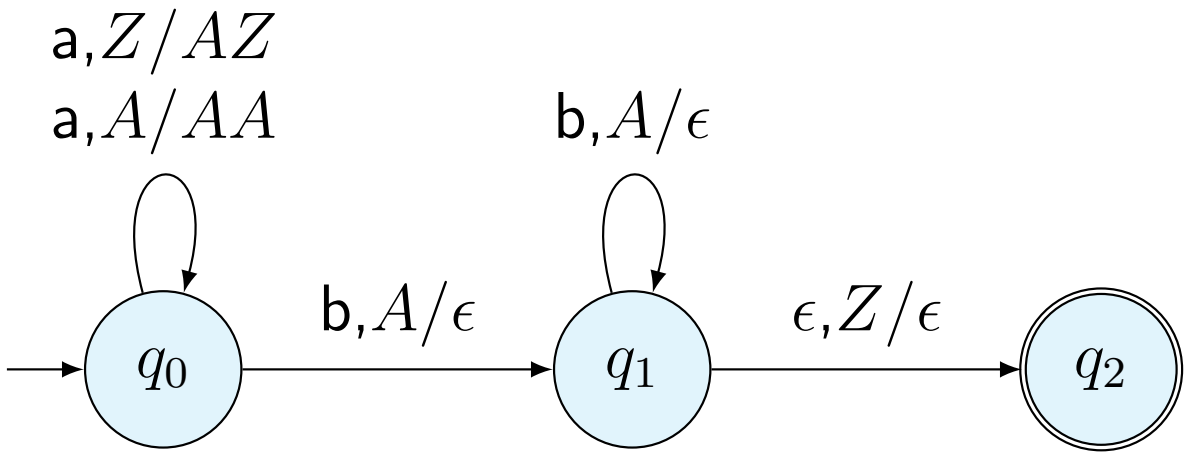
動作例:abaaccdc

$$
\begin{aligned}
(q_0, \mathsf{abaaddcd}, Z) &\vdash (q_0, \mathsf{baaddcd}, AZ) \\
&\vdash (q_0, \mathsf{aaddcd}, BAZ) \\
&\vdash (q_0, \mathsf{addcd}, ABAZ) \\
&\vdash (q_0, \mathsf{ddcd}, AABAZ) \\
&\vdash (q_2, \mathsf{dcd}, ABAZ) \\
&\vdash (q_2, \mathsf{cd}, BAZ) \\
&\vdash (q_2, \mathsf{d}, AZ) \\
&\vdash (q_2, \epsilon, Z) \\
&\vdash (q_3, \epsilon, \epsilon)
\end{aligned}
$$

動作例:ababdcdc

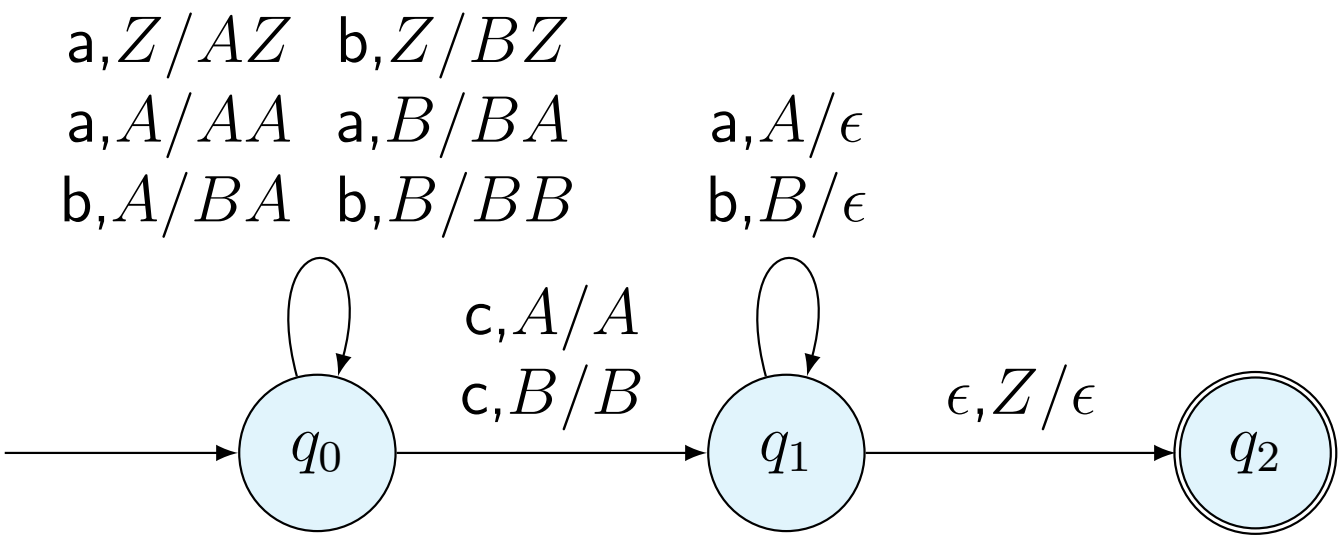
$$
\begin{aligned}
(q_0, \mathsf{ababdcdc}, Z) &\vdash (q_0, \mathsf{babdcdc}, AZ) \\
&\vdash (q_0, \mathsf{abdcdc}, BAZ) \\
&\vdash (q_0, \mathsf{bdcdc}, ABAZ) \\
&\vdash (q_0, \mathsf{dcdc}, BABAZ) \\
&\vdash (q_1, \mathsf{cdc}, ABAZ) \\
&\vdash (q_1, \mathsf{dc}, BAZ) \\
&\vdash (q_1, \mathsf{c}, AZ) \\
&\vdash (q_1, \epsilon, Z) \\
&\vdash (q_3, \epsilon, \epsilon)
\end{aligned}
$$

受理言語



- 入力とスタックが空になった時に、終状態に居るか？
- 例 2.1 では、$\left\{\mathrm{a}^i\mathrm{b}^i | i \in N\right\}$ を受理
  - a の数をスタック文字 A で記録
  - テープ上の b とスタック上の A を照合
  - FA では受理できない言語: 任意の数の a の「数」を記録
- 例 2.2 の受理言語は?
- 次の例 3.1 では、$\left\{wcw^R | w \in (\mathrm{a}+\mathrm{b})^*\right\}$ を受理

例 3.1:


a,Z/AZ b,Z/BZ
a,A/AA a,B/BA a,A/ϵ
b,A/BA b,B/BB b,B/ϵ
c,A/A
c,B/B
ϵ,Z/ϵ
q_0
q_1
q_2

$$Q = \{q_0, q_1, q_2\}, \qquad F = \{q_2\},$$
$$\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, \qquad \Gamma = \{A, B, Z\}.$$

$$\delta(q_0, \mathsf{a}, Z) = (q_0, AZ), \quad \delta(q_0, \mathsf{a}, A) = (q_0, AA), \quad \delta(q_0, \mathsf{a}, B) = (q_0, AB),$$
$$\delta(q_0, \mathsf{b}, Z) = (q_0, BZ), \quad \delta(q_0, \mathsf{b}, A) = (q_0, BA), \quad \delta(q_0, \mathsf{b}, B) = (q_0, BB),$$
$$\delta(q_0, \mathsf{c}, A) = (q_1, A), \quad \delta(q_0, \mathsf{c}, B) = (q_1, B),$$
$$\delta(q_1, \mathsf{a}, A) = (q_1, \epsilon), \quad \delta(q_1, \mathsf{b}, B) = (q_1, \epsilon), \quad \delta(q_1, \epsilon, Z) = (q_2, \epsilon).$$

動作例

$$
\begin{aligned}
(q_0, \mathsf{abaacaaba}, Z) &\vdash (q_0, \mathsf{baacaaba}, AZ) \\
&\vdash (q_0, \mathsf{aacaaba}, BAZ) \\
&\vdash (q_0, \mathsf{acaaba}, ABAZ) \\
&\vdash (q_0, \mathsf{caaba}, AABAZ) \\
&\vdash (q_1, \mathsf{aaba}, AABAZ) \\
&\vdash (q_1, \mathsf{aba}, ABAZ) \\
&\vdash (q_1, \mathsf{ba}, BAZ) \\
&\vdash (q_1, \mathsf{a}, AZ) \\
&\vdash (q_1, \epsilon, Z) \\
&\vdash (q_2, \epsilon, \epsilon)
\end{aligned}
$$

動作失敗例

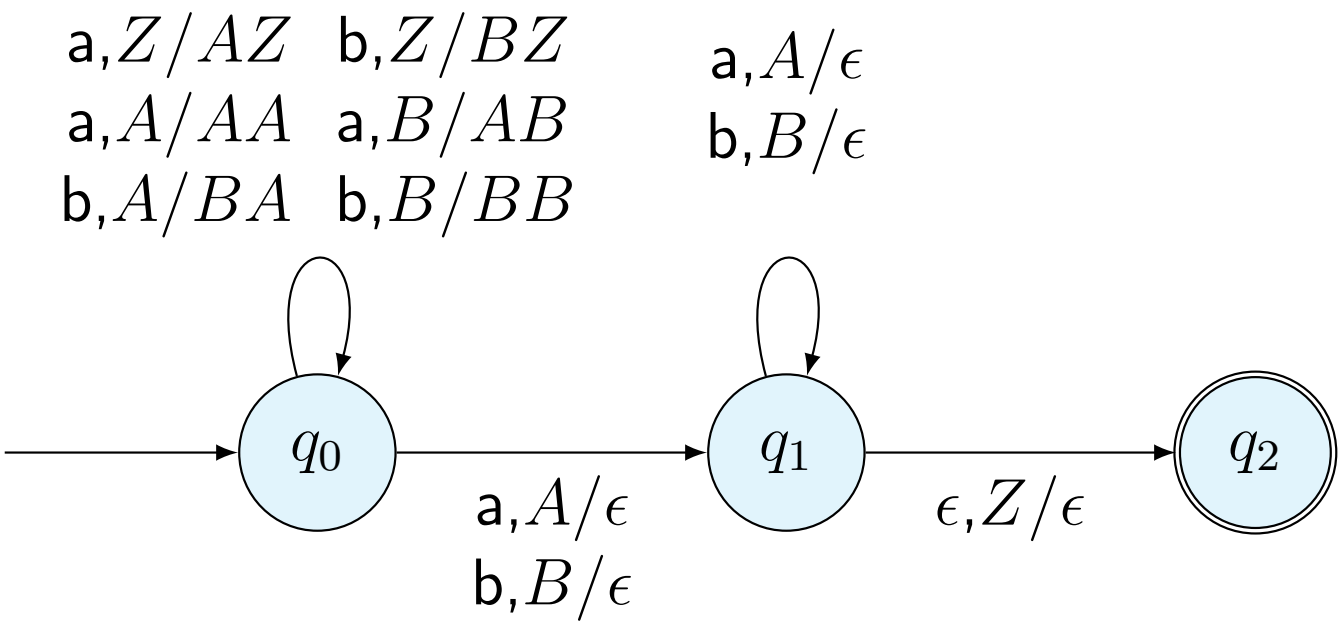
$$
\begin{aligned}
(q_0, \mathsf{abaaaaaba}, Z) &\vdash (q_0, \mathsf{baaaaba}, AZ) \\
&\vdash (q_0, \mathsf{aaaaba}, BAZ) \\
&\vdash (q_0, \mathsf{aaaba}, ABAZ) \\
&\vdash (q_0, \mathsf{aaba}, AABAZ) \\
&\vdash (q_0, \mathsf{aba}, AAABAZ) \\
&\vdash (q_0, \mathsf{ba}, AAAABAZ) \\
&\vdash (q_0, \mathsf{a}, BAAAABAZ) \\
&\vdash (q_0, \epsilon, ABAAAABAZ)
\end{aligned}
$$

非決定性プッシュダウンオートマトン
Non-deterministic PDA

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle \tag{4.1}$$

- Q: 内部状態の集合
- Σ: テープのアルファベット
- Γ: スタックのアルファベット
- $\delta : Q \times \Sigma \times \Gamma \to 2^{Q \times \Gamma^*}$: 遷移関数
- $q_0 \in Q$: 初期状態
- $Z_0 \in \Gamma$: スタックの底の記号
- $F \subseteq Q$: 終状態の集合

例 4.1:


a,Z/AZ
b,Z/BZ
a,A/AA
a,B/AB
b,A/BA
b,B/BB
a,A/ϵ
b,B/ϵ
q_0
q_1
q_2
a,A/ϵ
b,B/ϵ
ϵ,Z/ϵ

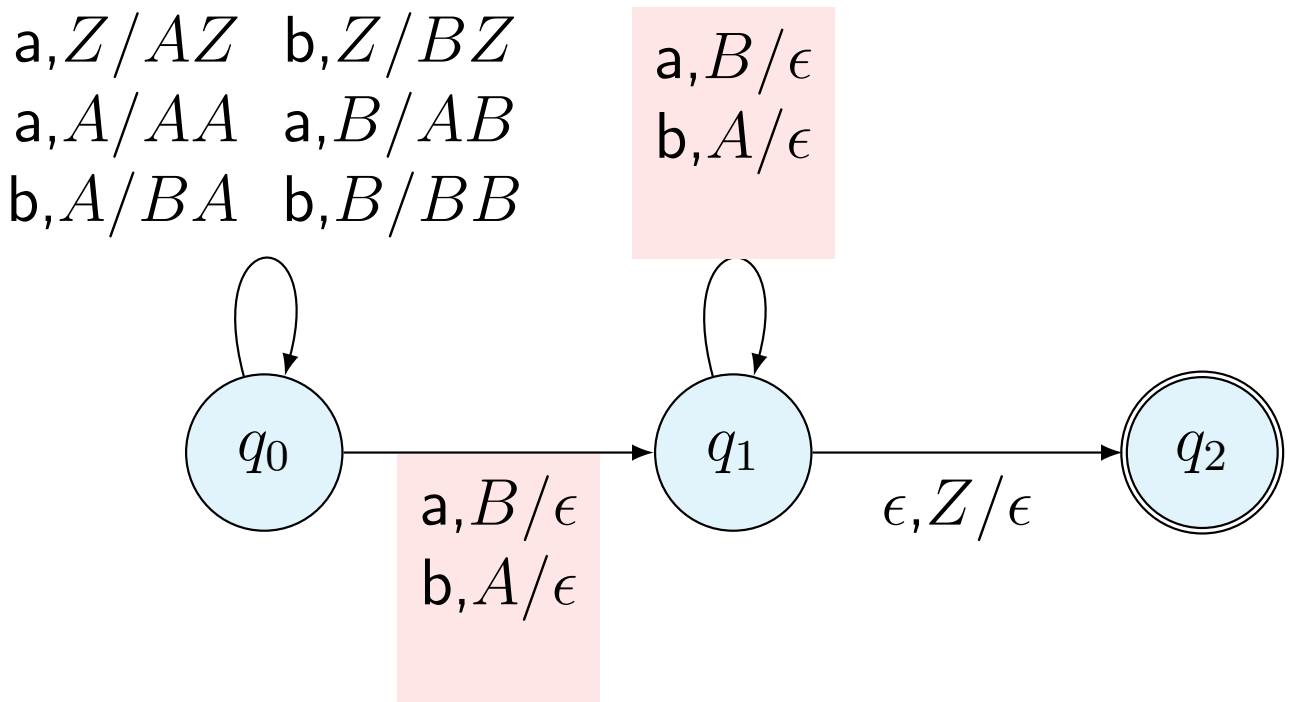
$$Q = \{q_0, q_1, q_2\}, \qquad F = \{q_2\},$$
$$\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, \qquad \Gamma = \{A, B, Z\}.$$

$$\delta(q_0, \mathsf{a}, Z) = \{(q_0, AZ)\} \qquad \delta(q_0, \mathsf{a}, A) = \{(q_0, AA), (q_1, \epsilon)\}$$
$$\delta(q_0, \mathsf{a}, B) = \{(q_0, AB)\} \qquad \delta(q_0, \mathsf{b}, Z) = \{(q_0, BZ)\}$$
$$\delta(q_0, \mathsf{b}, A) = \{(q_0, BA)\} \qquad \delta(q_0, \mathsf{b}, B) = \{(q_0, BB), (q_1, \epsilon)\}$$
$$\delta(q_1, \mathsf{a}, A) = \{(q_1, \epsilon)\} \qquad \delta(q_1, \mathsf{b}, B) = \{(q_1, \epsilon)\}$$
$$\delta(q_1, \epsilon, Z) = \{(q_2, \epsilon)\}$$

動作 (受理した例)

$$
\begin{aligned}
(q_0, \mathsf{abaaaaba}, Z) &\vdash (q_0, \mathsf{baaaaba}, AZ) \\
&\vdash (q_0, \mathsf{aaaaba}, BAZ) \\
&\vdash (q_0, \mathsf{aaaba}, ABAZ) \\
&\vdash (q_0, \mathsf{aaba}, AABAZ) \\
&\vdash (q_1, \mathsf{aba}, ABAZ) \\
&\vdash (q_1, \mathsf{ba}, BAZ) \\
&\vdash (q_1, \mathsf{a}, AZ) \\
&\vdash (q_1, \epsilon, Z) \\
&\vdash (q_1, \epsilon, \epsilon)
\end{aligned}
$$

例 4.2:


a,Z/AZ b,Z/BZ
a,A/AA a,B/AB
b,A/BA b,B/BB
a,B/ϵ
b,A/ϵ
q_0
q_1
q_2
a,B/ϵ
b,A/ϵ
ϵ,Z/ϵ

$$Q = \{q_0, q_1\}, \qquad F = \{q_1\},$$
$$\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, \qquad \Gamma = \{A, B, Z\}.$$

$$\delta(q_0, \mathsf{a}, Z) = \{(q_0, AZ)\} \qquad \delta(q_0, \mathsf{a}, A) = \{(q_0, AA)\}$$
$$\delta(q_0, \mathsf{a}, B) = \{(q_0, AB), (q_1, \epsilon)\} \qquad \delta(q_0, \mathsf{b}, Z) = \{(q_0, BZ)\}$$
$$\delta(q_0, \mathsf{b}, A) = \{(q_0, BA), (q_1, \epsilon)\} \qquad \delta(q_0, \mathsf{b}, B) = \{(q_0, BB)\}$$
$$\delta(q_1, \mathsf{a}, B) = \{(q_1, \epsilon)\} \qquad \delta(q_1, \mathsf{b}, A) = \{(q_1, \epsilon)\}$$
$$\delta(q_1, \epsilon, Z) = \{(q_2, \epsilon)\}$$

動作 (受理した例)

$$
\begin{aligned}
(q_0, \mathsf{abaabbbab}, Z) &\vdash (q_0, \mathsf{baabbbab}, AZ) \\
&\vdash (q_0, \mathsf{aabbbab}, BAZ) \\
&\vdash (q_0, \mathsf{abbbab}, ABAZ) \\
&\vdash (q_0, \mathsf{bbbab}, AABAZ) \\
&\vdash (q_1, \mathsf{bab}, ABAZ) \\
&\vdash (q_1, \mathsf{ab}, BAZ) \\
&\vdash (q_1, \mathsf{b}, AZ) \\
&\vdash (q_1, \epsilon, Z) \\
&\vdash (q_1, \epsilon, \epsilon)
\end{aligned}
$$

- 最初の例では、$\left\{ww^R | w \in (\mathsf{a} + \mathsf{b})^*\right\}$ を受理。折り返しの文字 c は不要
- 二番目の例では、w^R においてaとbを入れ替えた文字列を受理

PDA の受理言語

- PDA の受理言語は、正規表現では表せないもの
  - 前半と後半の文字数が同じ、前後を反転などは正規表現では表せない
  - $L_{01} = \{0^n 1^n | n \in N\}$ は正規表現で表せない: 反復補題
- スタックを使うことで、前半の文字列を覚えることができる
  - 長さに制限なし
- 再帰関数の実装にはスタックが必要

回文の集合は正規言語ではない

- 0 と 1 からなる回文全体 L が正規言語であると仮定
- 反復補題に現れる n に対して、$w = xyz = 0^n1^n1^n0^n$ を考える
- $|xy| \leq n$ より、$y = 0^k (1 \leq k \leq n)$

$$xy^0z = xz = 0^{n-k}1^n1^n0^n \notin L$$

- L は正規言語ではない

回文(palindrome)受理するPythonコード:再帰

```
1  def palindrome(inputStr:str) -> bool:
2      if len(inputStr) <= 1:
3          return True
4      if inputStr[0] != inputStr[-1]:
5          return False
6      return palindrome(inputStr[1:-1])
```

回文 (palindrome) 受理する Python コード: 非再帰

```
1  def palindrome2(inputStr:str) -> bool:
2      myStack:Stack[str] = Stack()
3      n = len(inputStr)
4      m = n//2
5      for i in range(m):#前半をスタックに積む
6          myStack.push(inputStr[i])
7      if n % 2 !=0:#入力の長さが奇数の場合
8          m += 1
9      for j in range(m,n):#後半と照合
10         c = myStack.pop()
11         if c != inputStr[j]:
12             return False
13     return True
```