期末試験のお知らせ

7月28日(月) 11:00 〜 12:30

1-106教室 (ここじゃない)

- 次回(7/21)の講義内容まで
- 学生証必携

レポート提出について

期日: 8月4日(月)20時頃まで

内容:

配布プリントのレポート問題の例のような内容、及び授業に関連する内容で、授業内容の理解または発展的な取組みをアピールできるようなもの

提出方法:

- **授業時に手渡し**
- **4-574 室扉のレポートポスト**
- **電子メイル**

定理:

L : **正規言語**

$\Updownarrow$

L **が或る決定性有限オートマトンで認識される**

$\Updownarrow$

L **が或る非決定性有限オートマトンで認識される**

非決定性有限オートマトンで認識できない言語が存在する!!

($\Longleftrightarrow$ 正規でない言語が存在する)

例: $A = \{a^n b^n \mid n \geq 0\}$

(a と b との個数が同じ)

証明は部屋割り論法

(の一種の pumping lemma)

による

より強力な計算モデルが必要

- **プッシュダウンオートマトン**
- **チューリングマシン**

例: “文法的に正しい” 数式とは どのようなものか?

- 単独の文字 (変数名) は式
- 式と式とを演算子で繋いだものは式
- 式を括弧で括ったものは式
- それだけ

$\longrightarrow$ これは式を作り出す規則とも考えられる

“文法的に正しい” 数式

初期記号 (開始変数) E から出発して、
次の規則のいづれかを
“非決定的に” 適用して得られるもの のみ

- $E \to A$
- $E \to EBE$
- $E \to (E)$
- $A \to$ 変数名のどれか
- $B \to$ 演算子のどれか
- 変数名・演算子・$(\ \cdot\ )$ は
それ以上書換えない (終端記号)

$\longrightarrow$ 生成規則 (書換規則)

生成規則を与えることでも
言語を定めることが出来る

$\longrightarrow$ **生成文法**

生成規則による “正しい” 語の生成

- **初期変数を書く**
- **今ある文字列中の或る変数を**
  生成規則のどれかで書換える
- **変数がなくなったら終わり**

例: $\{a^{2n}b^{2m+1}|n, m \geq 0\}$

(a が偶数個 (0 個も可) 続いた後に、b が奇数個続く)

正規表現で表すと、 $(aa)^*b(bb)^*$

- $S \to aaS$
- $S \to bB$
- $B \to bbB$
- $B \to \varepsilon$

まとめて次のようにも書く

- $S \to aaS|bB$
- $B \to bbB|\varepsilon$

実際の (自然言語を含めた) “文法” では、
或る特定の状況で現われた場合だけ
適用できる規則もあるだろう。

そのような生成規則は例えば次の形:

- $uAv \longrightarrow uwv$

$u, v \in \Sigma^*$ **: 文脈 (context)**

変数 A **が** uAv **の形で現われたら、**
語 $w \in \Sigma^*$ **で書換えることが出来る。**

生成文法の形式的定義

- V：有限集合 (**変数**の集合)
- Σ：有限集合 (**終端記号**の集合)
  ここに $V \cap \Sigma = \emptyset$
- R：有限集合 $\subset (V \cup \Sigma)^* \times (V \cup \Sigma)^*$
  (**規則**の集合)
- $S \in V$：**開始変数**

$(v, w) \in R$ が生成規則 $v \to w$ を表す。

文脈自由文法: 文脈が全て空列 ε

即ち、規則が全て $A \to w$ $(A \in V)$ の形

文脈自由文法の形式的定義

- V : 有限集合 (**変数**の集合)
- Σ : 有限集合 (**終端記号**の集合)
  ここに $V \cap \Sigma = \emptyset$
- R : 有限集合 $\subset V \times (V \cup \Sigma)^*$
  (**規則**の集合)
- $S \in V$: **開始変数**

$(A, w) \in R$ が生成規則 $A \to w$ を表す。

例: 言語 $A = \{a^n b^n | n \geq 0\}$ は
正規言語ではないが文脈自由言語である。

- $S \to aSb|\varepsilon$

従って、

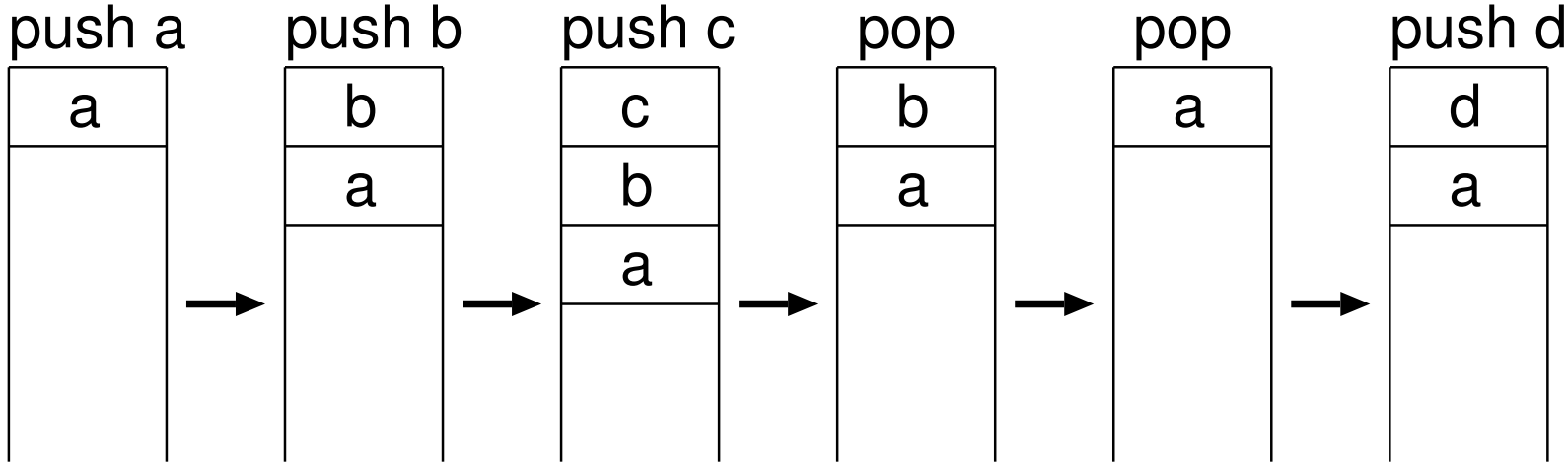
文脈自由言語は正規言語より真に広い!!

さて、正規言語を計算するモデルが
有限オートマトンであった。

文脈自由言語を計算するモデル
$\cdots$ 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

(非決定性) 有限オートマトンに
プッシュダウンスタックを取り付けたもの



無限 (非有界) の情報を保持できるが、
読み書きは先頭だけ
· · · LIFO (Last In First Out)

プッシュダウンオートマトンの形式的定義

$$M = (Q, \Sigma, \Gamma, \delta, s, F)$$

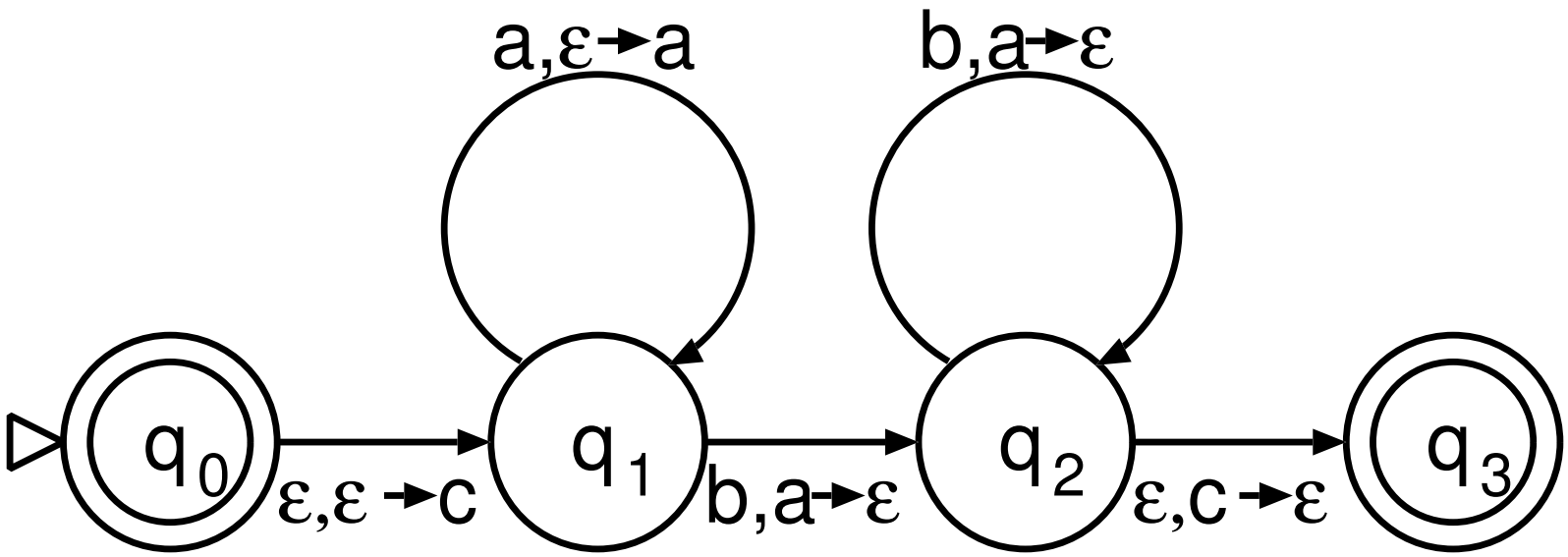
- Q：有限集合 $\cdots$ 状態の集合
- Σ：有限集合 $\cdots$ alphabet
- Γ：有限集合 $\cdots$ stack alphabet
  $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$ と置く。
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$
  ：遷移関数 $\cdots$ 可能な遷移先全体
- $s \in Q \cdots$ 初期状態
- $F \subset Q \cdots$ 受理状態の集合

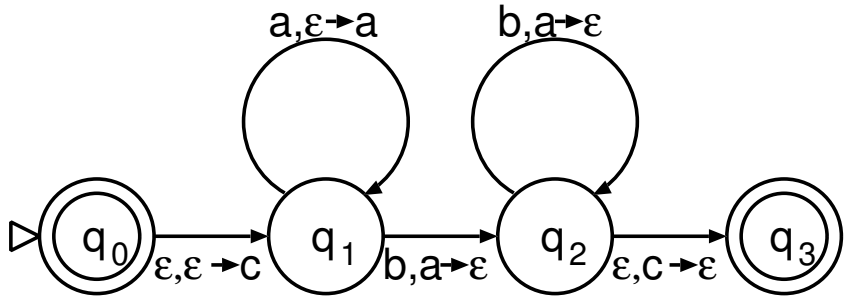
- $(r, y) \in \delta(q, a, x)$ **とは、**
  「入力 a **を読んだとき、**
  状態 q **でスタックの先頭が** x **なら、**
  スタックの先頭を y **に書換えて、**
  状態 r **に移って良い」**
  ということ (`pop; push y`**)**

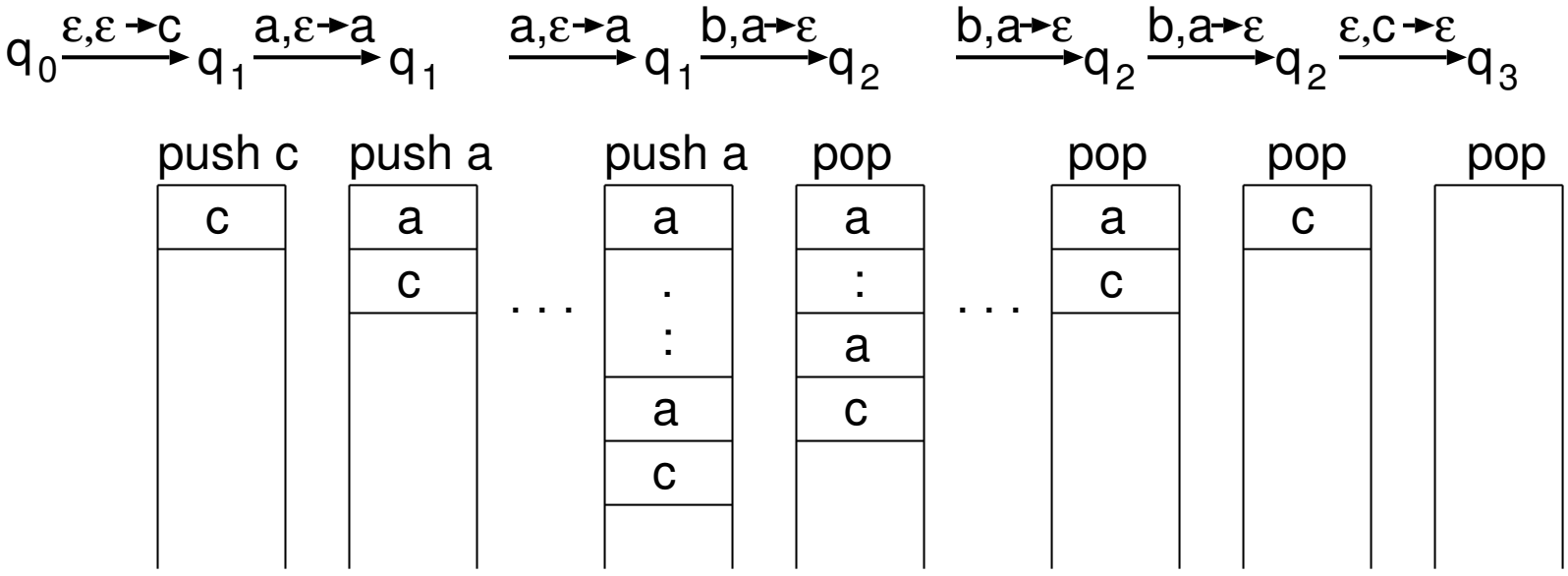
- $x = y$ **は書き換え無し**
- $x = \varepsilon$ **は push のみ**
- $y = \varepsilon$ **は pop のみ**
- $a = \varepsilon$ **は入力を読まずに遷移**

例: 言語 $A = \{a^n b^n | n \geq 0\}$ を認識する プッシュダウンオートマトン

$$\Sigma = \{a, b\}, \qquad \Gamma = \{a, b, c\}$$



PDA  による文字列 $a^n b^n$ の受理



定理:

L : **文脈自由言語**

$\Updownarrow$

L **が或るプッシュダウンオートマトンで認識される**

例: 回文全体の成す言語は文脈自由

- $S \to aSa|bSb|a|b|\varepsilon$

問: 回文全体の成す言語を受理する
プッシュダウンオートマトンを構成せよ。

例: 回文全体の成す言語は文脈自由

- $S \longrightarrow aSa|bSb|a|b|\varepsilon$

問: 回文全体の成す言語を受理する
プッシュダウンオートマトンを構成せよ。

例: 回文全体の成す言語は文脈自由

- $S \longrightarrow aSa|bSb|a|b|\varepsilon$

問: 回文全体の成す言語を受理する プッシュダウンオートマトンを構成せよ。

プッシュダウンオートマトンでは認識できない言語の例

同じ文字列 2 回の繰返しから成る文字列全体

$$A = \{ww | w \in \Sigma^*\}$$

入力を読み直せないのが弱点

$\longrightarrow$ より強力な計算モデルが必要

プッシュダウンオートマトンでは認識できない言語の例

同じ文字列 2 回の繰返しから成る文字列全体

$$A = \{ww | w \in \Sigma^*\}$$

入力を読み直せないのが弱点

$\longrightarrow$ より強力な計算モデルが必要

一つの方法としては、

入力を覚えておくために
プッシュダウンスタックをもう一つ
使えることにする。

実際これで
真により強い計算モデルが得られる。

しかし、通常はこれと同等な
次のような計算モデルを考える。
… チューリングマシン

一つの方法としては、

入力を覚えておくために
プッシュダウンスタックをもう一つ
使えることにする。

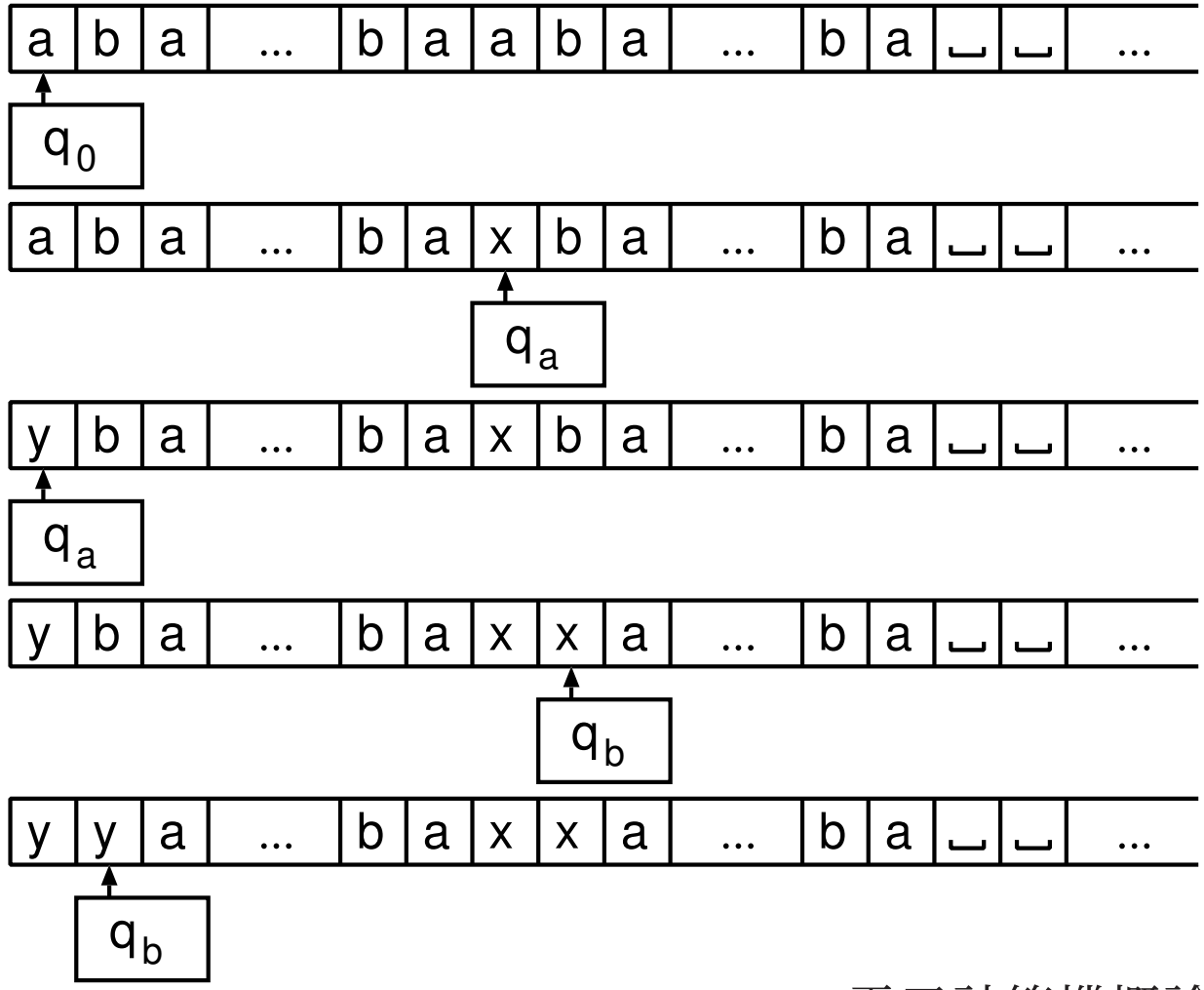
実際これで
真により強い計算モデルが得られる。

しかし、通常はこれと同等な
次のような計算モデルを考える。
... チューリングマシン

チューリングマシン

- 有限個の内部状態を持つ
- 入力データはテープ上に一区画一文字づつ書き込まれて与えられる
- データを読み書きするヘッドがテープ上を動く
- 遷移関数は次の形: 内部状態とヘッドが今いる場所の文字とによって、その場所の文字を書き換え、次の内部状態に移り、ヘッドを左か右かに動かす。
- 受理状態または拒否状態に達したら停止するが、停止しないこともある。

(非決定性) チューリングマシンによる言語 $A = \{ww | w \in \Sigma^*\}$ の認識



チューリングマシン T が言語 A を認識する

$$\Updownarrow$$

$$A = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{r} \textbf{入力 } w \textbf{ に対し、} \\ \textbf{受理状態で停止する} \\ \textbf{遷移が存在} \end{array} \right\}$$

$$\Updownarrow$$

$$w \in A \Longleftrightarrow \textbf{入力 } w \textbf{ に対し、受理状態で停止する遷移が存在}$$

チューリングマシン T が言語 A を判定する

$\Updownarrow$

T は A を認識し、
かつ、全ての入力に対し必ず停止する

$\Updownarrow$

$w \in A \Longleftrightarrow$ **入力 w に対し、**
受理状態で停止する遷移が存在

かつ

$w \notin A \Longleftrightarrow$ **入力 w に対し、**
拒否状態で停止する遷移が存在