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)

- 今日の講義内容まで
- 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)

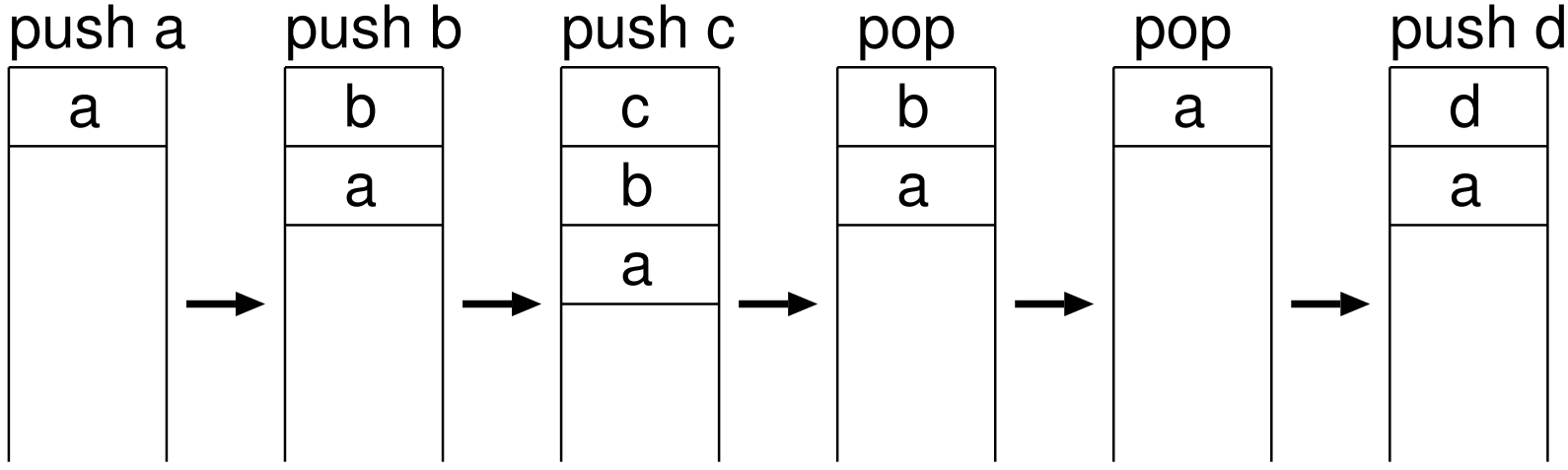
による

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**

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

$$M = (Q; \Sigma; \Gamma; \pm; s; F)$$

- 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
  $\Sigma_\varepsilon := \Sigma \cup \{''\}; \Gamma_\varepsilon := \Gamma \cup \{''\}$ と置く。
- $\pm : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$
  ：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

- $(r; y) \in \pm(q; a; x)$ **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**
- $x = ''$ **は push のみ**
- $y = ''$ **は pop のみ**
- $a = ''$ **は入力を読まずに遷移**

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

- $S \rightarrow aSa|bSb|a|b|''$

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

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。

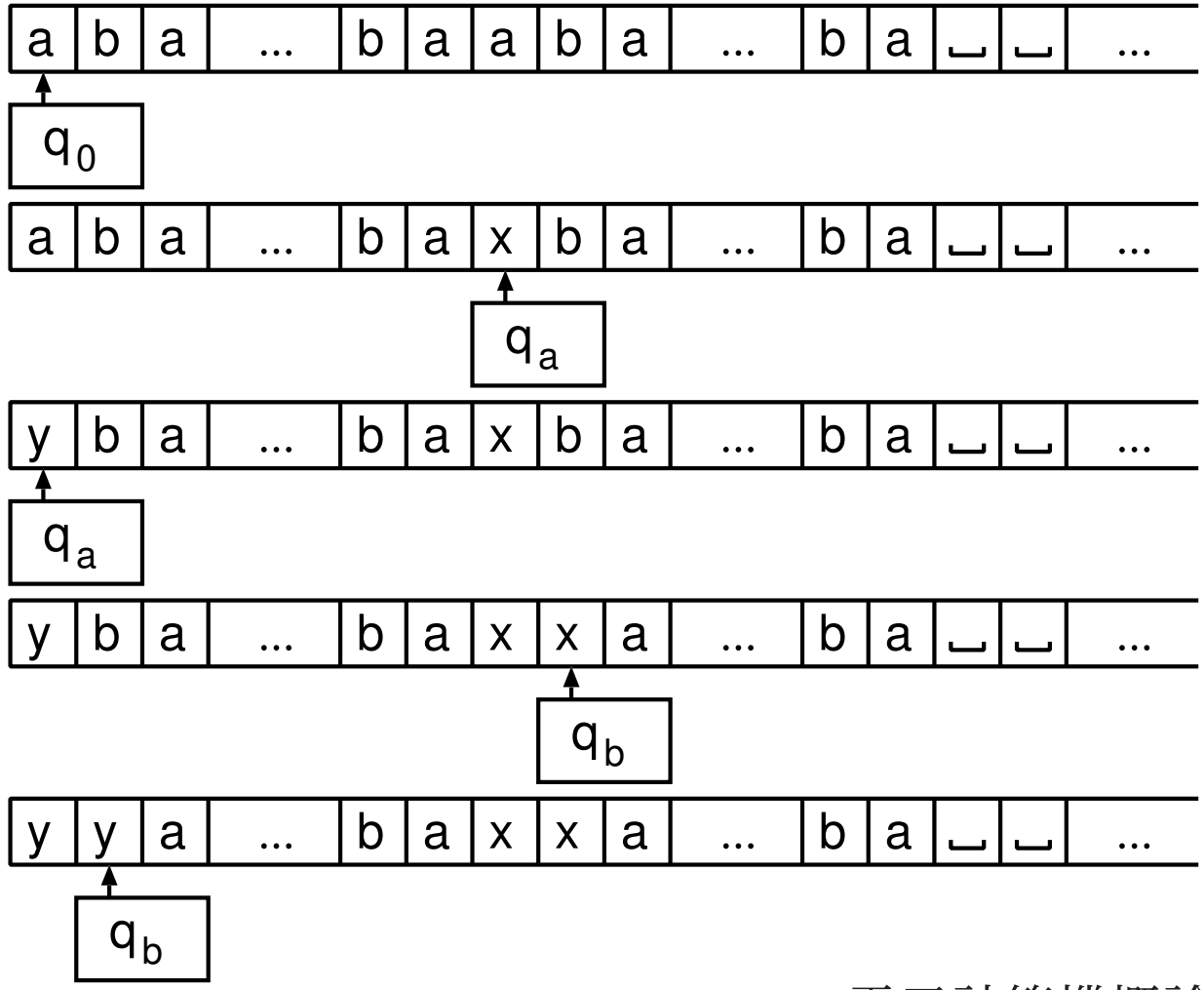
実際これで
真により強い計算モデルが得られる。

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

$$\Updownarrow$$

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

$$\Updownarrow$$

$w \in A \iff$ 入力 w 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

“Church-Turing の提唱”

「全てのアルゴリズム (計算手順) は、
チューリングマシンで実装できる」

(アルゴリズムと呼べるのは
チューリングマシンで実装できるものだけ)

. . . 「アルゴリズム」の定式化

“Church-Turing の提唱”

「全てのアルゴリズム (計算手順) は、
チューリングマシンで実装できる」

(アルゴリズムと呼べるのは
チューリングマシンで実装できるものだけ)

. . . 「アルゴリズム」の定式化

“Church-Turing の提唱”

「全てのアルゴリズム (計算手順) は、
チューリングマシンで実装できる」

(アルゴリズムと呼べるのは
チューリングマシンで実装できるものだけ)

. . . 「アルゴリズム」の定式化

プログラム内蔵方式 (von Neumann 型)

$\cdots$ プログラムもデータとして保持

$\longrightarrow$ 一つの機械で様々な計算を柔軟に実現

同様の働きをする

チューリングマシン U が構成できる

$\cdots$ 万能チューリングマシン

(universal Turing machine)

プログラム内蔵方式 (von Neumann 型)

$\cdots$ プログラムもデータとして保持

$\longrightarrow$ 一つの機械で様々な計算を柔軟に実現

同様の働きをする

チューリングマシン U が構成できる

$\cdots$ 万能チューリングマシン

(universal Turing machine)

プログラム内蔵方式 (von Neumann 型)

$\cdots$ プログラムもデータとして保持

$\longrightarrow$ 一つの機械で様々な計算を柔軟に実現

同様の働きをする

チューリングマシン U が構成できる

$\cdots$ 万能チューリングマシン

(universal Turing machine)

万能チューリングマシン

全てのチューリングマシンの動作を模倣する

入力: $(\langle M \rangle; w)$

- $\langle M \rangle$: **機械** M **の符号化**
  (プログラムに相当)
- w : M **に与える入力データ**

出力: 機械 M **が入力** w **を受理するかどうか**

定理

言語

$$A_{\mathtt{TM}} = \left\{ (\langle M \rangle; w) \,\middle|\, \begin{array}{l} \langle M \rangle : \text{TM } M \text{ の符号化} \\ M \text{ が入力 } w \text{ を受理} \end{array} \right\}$$

は認識可能だが、判定可能ではない**。**

証明は一種の対角線論法による。
(Russell のパラドックス風)

定理

言語

$$A_{\mathtt{TM}} = \left\{ (\langle M \rangle; w) \,\middle|\, \begin{array}{l} \langle M \rangle : \textbf{TM } M \textbf{ の符号化} \\ M \textbf{ が入力 } w \textbf{ を受理} \end{array} \right\}$$

は認識可能だが、判定可能ではない**。**

証明は一種の対角線論法による。
(Russell のパラドックス風)

$A_{\mathtt{TM}}$ を判定する TM U があったとする。

入力 $\langle M \rangle$ に対し、

- M が $\langle M \rangle$ を受理するなら拒否
- M が $\langle M \rangle$ を拒否するなら受理

となる TM D が (U を使って) 作れる。

これに、入力 $\langle D \rangle$ を喰わせよ。

$A_{\mathtt{TM}}$ を判定する TM U があったとする。

入力 $\langle M \rangle$ に対し、

- M が $\langle M \rangle$ を受理するなら拒否
- M が $\langle M \rangle$ を拒否するなら受理

となる TM D が (U を使って) 作れる。

これに、入力 $\langle D \rangle$ を喰わせよ。

$A_{\mathtt{TM}}$ を判定する TM U があったとする。

入力 $\langle M \rangle$ に対し、

- M が $\langle M \rangle$ を受理するなら拒否
- M が $\langle M \rangle$ を拒否するなら受理

となる TM D が (U を使って) 作れる。

これに、入力 $\langle D \rangle$ を喰わせよ。

さて、本講義最後の話題は、

計算量

について

計算量

- **時間計算量**: 計算に掛かるステップ数
  (TM での計算の遷移の回数)
- **空間計算量**: 計算に必要なメモリ量
  (TM での計算で使うテープの区画数)

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い。

入力データ長 n に対する
増加のオーダー (Landau の O-記号) で表す

問題を解くアルゴリズムによって決まる

$\cdots$ **アルゴリズムの計算量**

$\longrightarrow$ **アルゴリズムの効率 の評価**

問題の計算量:

その問題を解くアルゴリズムの計算量の 下限

最も効率良く解くと、どれ位で解けるか

$=$ **どうしてもどれ位必要か**

$=$ **どれ位難しい問題か**

$\longrightarrow$ **問題の難しさ の評価**

例:

- **加法:** $O(n)$
- **乗法:** $O(n^2)$

  かと思いきや $O(n \log n \log \log n)$

  (高速フーリエ変換 (FFT))

例:

- **加法:** $O(n)$

- **乗法:** $O(n^2)$
  かと思いきや $O(n \log n \log \log n)$
  (高速フーリエ変換 (FFT))

重要な難しさのクラス:

多項式時間 P $\cdots \exists k : O(n^k)$

“事実上計算可能な難しさ”

大体
「しらみつぶし」が入ると
$O(2^n)$ **程度以上になる (指数時間 EXP)**
“事実上計算不可能”

例: 素数判定 (PRIMES)

試行除算 (小さい方から割っていく) だと
$O(n^k 2^{n/2})$ くらい掛かりそう

実は多項式時間で解ける!!

Agrawal-Kayal-Saxena
"PRIMES is in P" (2002)
(出版は
Ann. of Math. 160(2) (2004),781-793.)

例: 素数判定 (PRIMES)

試行除算 (小さい方から割っていく) だと
$O(n^k 2^{n/2})$ **くらい掛かりそう**

実は多項式時間で解ける!!

Agrawal-Kayal-Saxena
"PRIMES is in P" (2002)
(出版は
Ann. of Math. 160(2) (2004),781-793.)

このような効率の良い素数判定は、
具体的に素因数を見付けている訳ではない。

素因数分解は P であるかどうか未解決
(多項式時間アルゴリズムが知られていない)

素因数分解の困難さを利用した暗号方式
$\cdots$ **RSA 暗号** (Rivest-Shamir-Adleman)

鍵となる整数 n の素因数分解を
知っていれば解読できるが、
知らないと解読できない。

$\longrightarrow$ 来年春期の「応用数学 I」で

このような効率の良い素数判定は、
具体的に素因数を見付けている訳ではない。

素因数分解は P であるかどうか未解決
(多項式時間アルゴリズムが知られていない)

素因数分解の困難さを利用した暗号方式
$\cdots$ **RSA 暗号** (Rivest-Shamir-Adleman)

鍵となる整数 n の素因数分解を
知っていれば解読できるが、
知らないと解読できない。

$\longrightarrow$ 来年春期の「応用数学 I」で

計算量にも “非決定性” の概念がある

あてずっぽうを許して、
うまくいけばどの位で解けるか
= 答を知って、その検証にどの位かかるか

非決定性多項式時間 (NP):
非決定性の計算モデルで多項式時間で解ける

例: 素因数分解は NP
… 素因数を知っていれば割算 1 回だけ

未解決問題 (P vs NP Problem)

$$\mathbf{P} = \mathbf{NP}$$

であるか否か ？

“The Millennium Problems”
の 7 つの問題のうちの 1 つ
(賞金 $1M)

未解決問題 (P vs NP Problem)

$$\mathbf{P} = \mathbf{NP}$$

であるか否か ？

“The Millennium Problems”
の 7 つの問題のうちの 1 つ
(賞金 $1M)

秋期「電子計算機概論 II」について

- **担当: 谷口 肇 先生**
- **内容 (予定):**
  - **コンパイラの実装について**
  - **構文解析など**

引続いての履修をお奨めします。

おしまい