演習問題

簡易モデル上でのアセンブリ言語により、

自然数 n に対して、

1 から n までの和

を計算するプログラムを作成せよ

- 実行前 (入力): ラベル `A` の場所に
  数値 n が書き込まれている
- 実行後 (出力): ラベル `C` の場所に
  計算結果の数値が書き込まれている

正しく計算を行なうプログラムであっても
良し悪しがある

プログラム (アルゴリズム) の評価

- 計算時間が速い

(= 計算ステップ数が少ない)

$\cdots$ 時間計算量

- 使用メモリ量が少ない $\cdots$ 空間計算量

最低限どれだけ必要か

$\longrightarrow$ "問題の難しさ" の評価

$\cdots$ 計算量の理論

演習問題

簡易モデル上でのアセンブリ言語により、
自然数 n に対して、

1 から n までの和
を計算するプログラムを作成せよ

- **実行前 (入力): ラベル `A` の場所に**
  数値 n が書き込まれている
- **実行後 (出力): ラベル `C` の場所に**
  計算結果の数値が書き込まれている

  - **メモリ量: 12 words**
  - **ステップ数: $(7n + 5)$ steps**

まで絞れるようだ $\longrightarrow$ 挑戦者求む

但し。初期のプログラム開発では、確かに

- **少しでも使用メモリが少ない**
- **少しでも早い (ステップ数が少ない)**

プログラムが良いとされたが、

現在は、実際には、

- **移植性 (可搬性)・拡張性・部品性が高い**
- **開発・保守・管理コストが低い**

プログラムが求められている

背景:

- **ハードウェアコストの低下**
- **人的コストの (相対的) 上昇**
- **プロジェクト規模の拡大**

計算の理論

命令の実行 (= 「計算」) とは、

レジスタまたは主記憶の
現在の値 (状態) に従って、

その値を変更 (書込) すること

であった

計算の理論

プログラム内蔵方式 (von Neumann 型) では、

プログラム・データを区別なく
メモリ上に置いていたが、

プログラムとデータとは、やはり本質的に違う

- **プログラム: 一つの問題では固定**
- **データ: 可変な入力**

$\Downarrow$

どんな (有効な) データ (入力) が来ても、
所定の出力を返すことが要請される

計算の理論

或る問題の「計算が可能」

$\Updownarrow$

その計算を行なうプログラムが存在

$\Downarrow$

計算機の機能 ($=$「計算」のモデル)
を決めて議論する

ここでは、代表的な「計算のモデル」を
幾つか紹介する

問題を「計算する」とは

入力 (データ)

$\Downarrow$

$\boxed{\text{プログラム}}$

$\Downarrow$

出力・動作

原理・理論を考える際には、
出力は最も単純に「0 か 1 か」とする

- 0 : **拒否 (reject)**
- 1 : **受理 (accept)**

「問題」とは

入力 (データ)

⇓

$\boxed{\text{プログラム}}$

⇓

受理か拒否か

解くべき「問題」：入力を受理する条件

「問題」の例

入力の範囲: 文字 a, b から成る文字列

「問題」: 入力を受理する条件

- a **と** b **との個数が同じ**
- a **が幾つか続いた後に** b **が幾つか続いたもの**
- a **で始まり** a, b **が交互に並んで** b **で終わる**
- **同じ文字列** 2 **回の繰返しから成る**
- **回文 (palindrome)**

などなど

「問題」とは

それぞれの「問題」に対し、
定められた計算モデルで、
受理／拒否判定が可能 (問題が解ける) か ?

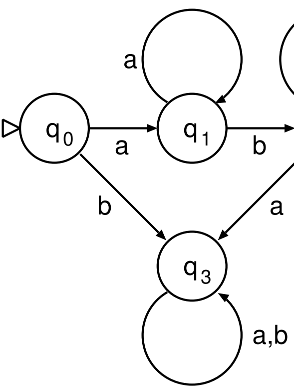
受理される文字列が
「文法的に正しい」文字列だと思えば、

「問題」とは「文法 (言語)」である

「文法的に正しい」かどうかの判定
· · · 「構文解析 (syntactic analysis)」

トン
計算モデル
械
,
‾nite automaton
)

有限オートマトンの例 (状態遷移図による表示)



有限オートマトンの形式的定義

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

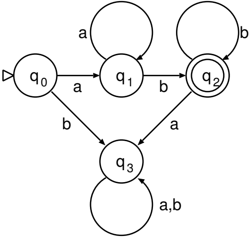
ここに、

- Q ： 有限集合 $\cdots$ **状態**の集合
- Σ ： 有限集合 $\cdots$ 入力文字の集合

  \alphabet"

- $\delta : Q \times \Sigma \to Q$ ： **遷移関数**
- $s \in Q$ $\cdots$ **初期状態**
- $F \subset Q$ $\cdots$ **受理状態**の集合

先の例



では、

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- $\delta : Q \times \Sigma \rightarrow \Sigma$:

	q_0	q_1	q_2	q_3
a	q_1	q_1	q_3	q_3
b	q_3	q_2	q_2	q_3

- $s = q_0 \in Q$
- $F = \{q_2\} \subset Q$

語・言語

Σ : 入力文字の有限集合 $\cdots$ **alphabet**

入力は Σ の元の有限列 (**語**, **word**)

$$w = a_1 a_2 \cdots a_n \;\; (a_i \in \Sigma)$$

その全体 $\Sigma^{¤}$

$$\Sigma^{¤} := \bigcup_{n=0}^{1} \Sigma^n \qquad (\Sigma^0 = \{\varepsilon\} : \text{空列})$$

言語 (**language**) : $\Sigma^{¤}$ の部分集合

言語 $A \subset \Sigma^{¤}$ に属する語 $w \in A$

$\cdots$ 言語 A に於いて \文法的に正しい"

有限オートマトンによる語の受理

有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ **が**
語 $w = a_1 a_2 \cdots a_n$ **を受理 (accept) する**

$$\Updownarrow$$

$\exists r_0, r_1, \ldots, r_n \in Q$:

- $r_0 = s$
- $\delta(r_{i ¡ 1}, a_i) = r_i$ **(**$i = 1, \ldots, n$**)**
- $r_n \in F$

$L(M)$ **:** M **が受理する語の全体** $\subset \Sigma^{¤}$
$\cdots$ M **が認識 (recognize) する言語**

M **は言語** $L(M)$ **の \文法" で、**
M **が受理する語は \文法的に正しい"**

演習問題

$\Sigma = \{a, b\}$ **とする。**

次の言語を認識する有限オートマトンを構成し、状態遷移図で表せ

(1) $A = \{a^{2n}b^{2m+1} | n, m \geq 0\}$
(a **が偶数個 (** 0 **個も可) 続いた後に、** b **が奇数個続く)**

(2) $B = \{vabbaaw | v, w \in \Sigma^*\}$
(部分列として $abbaa$ **を含む)**

演習問題

ちょっとしたコツ (tips):

「後に続く文字列が何だったら受理か」

が全く同じ状態は一つの状態にまとめられる。

これが違う状態はまとめられない。
(違う状態として用意する必要あり)

有限オートマトンでの計算可能性問題

- 言語 $A \subset \Sigma^*$ に対し、
  A を認識する有限オートマトン M
  が存在するか？

- 有限オートマトンによって
  認識可能な言語はどのようなものか？

$\longrightarrow$ **正規言語・正規表現**